O Método da Têmpera Simulada

Nelson Hein
Maria Salett Biembengut

1. INTRODUÇÃO

A têmpera simulada é uma família de métodos que, geralmente, dão respostas boas a problemas de otimização combinatorial. O uso de algoritmos determinísticos na resolução desta classe de problemas (ex.: o problema do caixeiro viajante) é geralmente impossível, devido ao tempo computacional que cresce em ordem fatorial ao se elevar as dimensões do problema, visto que o espaço do problema não pode ser explorado exaustivamente em tempo real.

Por esta razão uma boa heurística necessitava ser descoberta. A idéia da têmpera simulada (*Simulated Annealing*) vem da analogia entre este tipo de problemas e a termodinâmica.

2. A TÊMPERA EM SÓLIDOS

Este processo consiste em incrementar temperaturas de um metal até seu ponto de fusão, e a partir daí lentamente resfriado, para que o sólido obtido tenha uma estrutura muito bem organizada.

Em analogia com da têmpera de aços, sabe-se que após a temperatura de fusão T_0, as moléculas movem-se randomicamente. O movimento completamente livre das moléculas significa um estado de grande energia.

O metal é então resfriado da temperatura T_0 para T_1 (as temperaturas T_1 e T_0 são muito próximas). O tempo de resfriamento deve ser suficiente para que um equilíbrio térmico seja alcançado. Entremente, o nível de energia das moléculas decai com a temperatura.

Esta operação é repetida seguidas vezes, até que alcance a temperatura ambiente, resultando numa estrutura organizada, e um nível baixo de energia alcançado. Teoricamente, sabe-se que o "Estado Fundamental" da matéria é alcançado em 0 Kelvin (aprox. –273°C), ou seja é nesta temperatura que o estado de energia das moléculas é nulo.

A têmpera simulada é a combinação de dois processos:

– um processo que consiste em simular a evolução do sistema molecular para o estado de equilíbrio térmico;

– um processo de resfriamento, consistindo em baixar a temperatura do sistema molecular, levando-o ao equilíbrio.

3. A TÊMPERA SIMULADA NA OTIMIZAÇÃO

Muitos problemas podem ser formulados como segue:

$$\min F(\mathbf{x})$$
$$x \in X$$

Onde X é o conjunto de todas as soluções admissíveis do problema e F(**x**) é a função objetivo em relação a **x** [HEIN94].

Assumindo a analogia entre a função de energia no sistema físico e a função objetivo do problema, observa-se que o estado mínimo de energia do sistema físico é análogo à determinação da solução (ou soluções) do problema de otimização. Assim, o algoritmo desenvolvido para sistemas físicos pode ser transposto para problemas de otimização.

Nelson Hein é Mestre em Engenharia de Produção e Sistemas (UFSC), Doutorando em Engenharia de produção e Sistemas (UFSC), Professor da Universidade Regional de Blumenau (FURB) – Departamento de Matemática - Centro de Ciências Exatas e Naturais (CCEN)

Maria Salett Biembengut é Doutora em Engenharia de Produção e Sistemas (UFSC), Professora da Universidade Regional de Blumenau (FURB) – Departamento de Matemática - Centro de Ciências Exatas e Naturais (CCEN)

4. A HEURÍSTICA DA TÊMPERA SIMULADA

As heurísticas clássicas de otimização partem de um valor inicial que atenda a viabilidade do problema. Sobre este ponto inicial aplica-se então um algoritmo. Caso esse algoritmo mova a resposta para um ponto melhor, ou seja, apresente um valor menor para a função objetivo, então, ele é classificado como um algoritmo de otimização, e o processo é repetido até que o método não apresente mais melhorias sensíveis para a função objetivo, quando estabelecido uma tolerância aproximativa à solução ótima. Com isto corre-se o risco de finalizar num mínimo local da função objetivo. E assim todo ponto obtido nas proximidades deste ótimo local, em iterações posteriores, ocasiona um aumento no valor da função objetivo, impedindo o algoritmo de encaminhar-se ao ótimo global [WEA94]. Uma alternativa é aceitar valores que piorem temporariamente o valor da função objetivo, conforme pode-se verificar na figura 01, que apresenta o algoritmo da têmpera simulada.

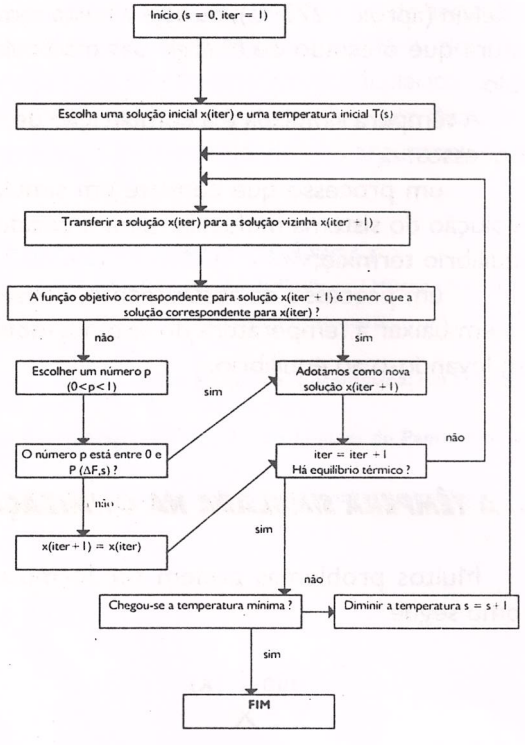

Fig. 02. O algoritmo da têmpera simulada

A têmpera simulada aceita novas soluções que contribuam em reduzir a função objetivo; por outro lado, se a nova solução for pior, ela poderá ser aceita com certa probabilidade P(DF,s), controlada pelo incremento na função objetivo DF e por s, que representa o número de resfriamentos realizados até então. Esta probabilidade deve proporcionar ao algoritmo, nas primeiras iterações, que ele aceite quase todas as soluções, piores ou melhores. E conforme o processo se desenvolve - com o abaixamento da temperatura - esta probabilidade em aceitar soluções piores é reduzida, com o propósito de estabilizar a solução do problema no seu ótimo global. O esquema da figura 02 apresenta o esboço da evolução espacial de uma função, utilizando a têmpera simulada.

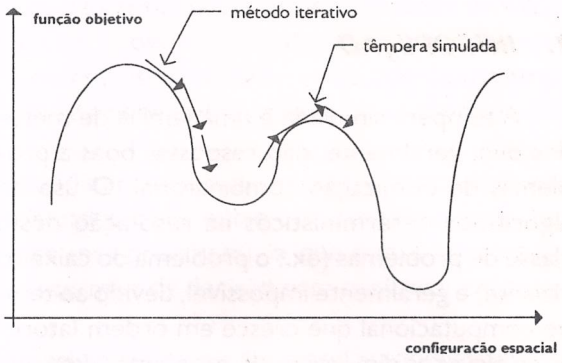

Fig.01. Evolução na configuração espacial

5. MUDANÇA DE PARÂMETROS

5.1. Mudança da probabilidade P(ΔF,s)

Em analogia com a física-estatística, ou ainda, mecânica-estatística, nós associamos para P a distribuição de probabilidade de Gibbs-Boltzmann [AAR90]:

$$P(\Delta F,s) = e^{[-\Delta F / T(s)]}$$

5.2. Os resfriamentos

No início do algoritmo, aconselha-se tomar uma temperatura alta (ex.: $T_0 = 100$), pois a probabilidade de aceitar uma configuração pior é muito importante. O fator $e^{[-\Delta F / T(s)]}$ é em geral mais próximo a 1 do que de 0. Este é o motivo pelo qual aceita-se, inicialmente, quase todas as configurações, que fazem com que o sistema evolua randomicamente pelo espaço de configurações.

Conforme a temperatura decai, a probabilidade em aceitar uma resposta pior, na configuração, também decai. E conforme a temperatura cai, o algoritmo torna-se cada vez mais exigente. Na medida em que a temperatura baixa em direção ao equilíbrio térmico (T_f=0+e), as probabilidades são diminuídas de forma que a freqüência

em aceitar soluções que apenas melhorem o valor da função objetivo se tornam predominantes, diminuindo assim o efeito perturbador da aleatoriedade gerada pela energia da têmpera.

Pela teoria das Cadeias de Markov, existem teoremas [AAR90] que comprovam que, sob certas hipóteses, o algoritmo da têmpera simulada converge para a configuração correspondente ao mínimo da função objetivo, com probabilidade igual a 1.

Porém na prática, uma hipótese necessária na teoria de Markov não pode ser atendida, pois o decaimento da temperatura deveria se dar muito lentamente, e isto traria problemas de tempo computacional, que não seria nada razoável.

Entretanto, caso seja feito uma dependência apropriada entre T e s, o algoritmo dirige-se a bons resultados, próximos ao ótimo. Em geral, T(s) é dado como por uma função degrau (fig.03).

A temperatura inicial T_0 é mantida durante A iterações no algoritmo. A segunda temperatura vale T_1 ($T_1 = \alpha T_0$ onde $0 < \alpha < 1$) que será usada durante A novas iterações. No s-ésimo passo teremos: $T_s = \alpha^s T_0$.

A temperatura T_s é definida por três parâmetros:

- T_0: temperatura inicial;
- α: valor entre 0 e 1;
- A: tamanho do passo (número de iterações durante o qual a temperatura é constante).

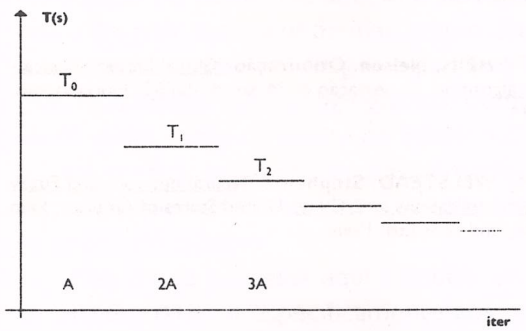

Fig.03. Evolução das temperaturas

T_0 deve ser suficientemente grande para que o sistema explore livremente o espaço de configurações, ou seja, sem que haja inicialmente grandes cobranças de melhoras. Com a próximo a 1 os movimentos são livres e o decaimento é lento. O valor de α deve ser suficientemente grande para que explore suficientemente soluções vizinhas.

5.3. Critério de parada

O algoritmo termina quando a taxa de decrescimento da função for menor que certa porcentagem inicialmente estipulada, durante um período de tempo, também demarcado anteriormente (geralmente longo), ou ainda por uma vizinhança (tolerância) centrada no ponto de equilíbrio térmico e raio e.

Tanto T_0, α, A e o critério de parada são determinados por tentativa e erro para cada problema em particular.

6. UM PROGRAMA EM BASIC

Este programa foi desenvolvido com base no algoritmo apresentado no item (4). A linguagem de programação usada foi o BASIC. O organizador do programa foi o professor Nelson Hein, o qual, segundo ele mesmo, faz parte do rol dos "programadores de final de semana". Fica portanto aos "experts" em programação, a oportunidade de desenvolver o mesmo programa na linguagem que mais lhes convier.

6.1. O Programa

```
rem "****************************************"
rem "************ TEMPERA SIMULADA ***********"
rem "************** OTIMIZACAO ***************"
rem "*************  ARTIGO - CEFET  **********"
rem "Autor: Prof. Nelson Hein - MTM-CCEN/FURB "
rem "****************************************"
1 cls
10 input "temperatura inicial = ";t
15 lprint "temperatura inicial = ";t
20 input "Aproximacao inicial = ";x(1)
25 lprint "Aproximacao inicial = ";x(1)
30 input "No. de iteracoes por tempera = ";n
35 lprint "No. de iteracoes por tempera = ";n
36 lprint
39 i=0
40 i=i+1
41 cls
45 print "temperatura atual = ";t
46 print "iteracao atual = ";i
47 print
50 y(1)=x(1)^2-x(1)+sin(3*x(1)^2+1)*(cos(2*x(1)))^2
60 print " x(1)= ";x(1):print" y(1)= ";y(1)
70 m=rnd(i)
80 if m<=0.5 then 90 else 110
90 x(2)=x(1)-t*m
100 goto 120
110 x(2)=x(1)+t*m
120y(2)=x(2)^2 x(2)+sin(3*x(2)^2+1)*(cos(2*x(2)))^2
130 if y(2)<y(1) then 140 else 170
140 print " x(2)= ";x(2):print" y(2)= ";y(2)
141 print:print:print
145 x(1)=x(2)
150 rem input "deseja continuar (s/n)= ";n$
160 rem if n$<>"n" then 41 else 1000
170 p=rnd(i+1)
180 dy=abs(y(2)-y(1))
190 b=exp(-dy/t)
200 if p<b then 210 else 220
210 x(1)=x(2)
220 if i=n then 230 else 40
230 t=t/1.01
240 if t<0.01 then 1000
250 goto 39
```

```
1000 lprint " x(final)= ";x(1)
1010 lprint " y(final)= ";y(1)
1020 lprint " temp. final = ";t
1030 lprint :lprint
1040 end
```

Observação: nas linhas 90 e 110 foi utilizado um regulador de passo, ou seja, conforme a temperatura cai, o passo randômico diminui de grandeza. Ele pode ser excluído. Porém com o regulador de passo, chega-se mais próximo ao ótimo (local ou global), frente a uma tolerância inicialmente imposta.

6.2. Alguns resultados

Inicialmente otimizou-se (minimizou-se) a função $y=x^4-4x^2-x$ (fig.04). Foram feitas 3 simulações, com diferente número de iterações por têmpera, obtendo-se os seguintes resultados:

Temperatura inicial =	10	10	10
Aproximação inicial =	-5	-5	-5
Nº de iter./têmpera =	3	5	10
x(final) =	1.513095	1.530343	1.534081
y(final) =	-5.43318	-5.410487	-5.408391

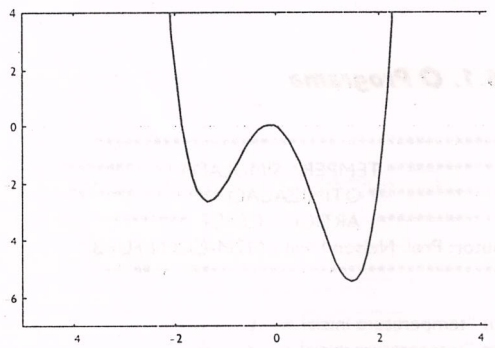

Fig.04.Gráfico da função $y=x^4-4x^2-x$

Usando o algoritmo de otimização global Hein-Stange [HEIN94], obtém-se o seguinte resultado:

$$x(final) = 1.473 \quad e \quad y(final) = -5.444$$

Outra função (minimizada) testada foi:

$$y=x^2-x+sen(3x^2+1)cos^2(2x)$$

cujo gráfico é da forma abaixo:

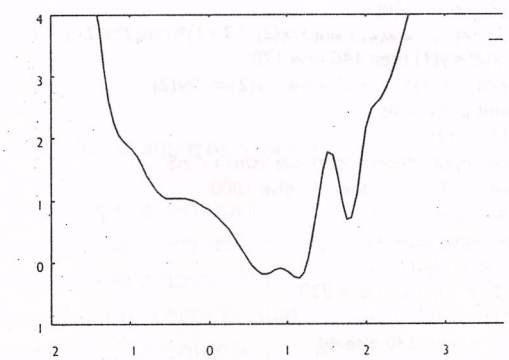

Fig.05. Gráfico da função $y=x^2-x+sen(3x^2+1)cos^2(2x)$

Os resultados aqui alcançados foram os seguintes:

Temperatura inicial =	10	10	10
Aproximação inicial =	-5	-5	-5
Nº de iter./têmpera =	3	5	10
x(final) = 1.78155	1.809433	1.809433	
y(final) = 0.6540099	0.6539383	0.6893461	

Porém, usando o algoritmo de otimização global chegamos a:

$$x(final) = 1.145 \text{ e } y(final) = -0.251$$

Assim foi necessário refazer a têmpera simulada:

Temperatura inicial = 10
Aproximação inicial = -5
Nº de iter./têmpera = 100
x(final) = 1.183309
y(final) = -0.2338665

Observação: o número de iterações por têmpera foi aumentado, permitindo que o algoritmo explorasse melhor o espaço de soluções. Isto serve de alerta, pois quanto mais complexo for o espaço de configuração, mais demorada deverá ser a têmpera simulada.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[AAR90] **AARTS, Emile e KORST, Jan.** Simulated Annealing and Boltzmann Machines: a Stochastic approach to combinatória optimization and neural computing. Great Britain. John Wiley and Sons Ltd, 1990.

[HEIN94] **HEIN, Nelson. Otimização** Global Determinística - Um algoritmo. Dissertação de Mestrado-UFSC, Florianópolis, 1994.

[WEL94] **WELSTEAD, Stephen T.** Neural network and Fuzzy Logic Applications in C/C++. United States of America, John Wiley and Sons Ltd, 1994.