Aspectos teóricos de simulated annealing e um algoritmo duas fases em otimização global[1]

Gabriel Haeser[2], Márcia A. Gomes–Ruggiero[3]
Departamento de Matemática Aplicada, IMECC, UNICAMP, 13083-859, Campinas, SP, Brasil.

Resumo. Neste trabalho descrevemos a teoria para *simulated annealing* e propomos um método híbrido para otimização global de problemas canalizados contínuos que em uma fase heurística utiliza *simulated annealing* e em uma fase local o solver GENCAN.

1. Introdução

Em pesquisas na literatura especializada sobre aplicações de técnicas em otimização contínua, principalmente na área de Engenharia Química, é possível constatar que um grande número de problemas resulta em modelos não lineares com funções não convexas. As não convexidades levam a múltiplos ótimos locais tornando difícil a tarefa de identificar o ótimo global, que é a solução de interesse nestas aplicações. A dificuldade central da busca pelo ótimo global resulta do fato de que os algoritmos usuais em otimização dependem fortemente do ponto inicial. No caso de convergência, é obtido um ponto estacionário, que pode não ser ótimo local do problema, muito menos ótimo global.

Descrevemos a teoria para *simulated annealing* [1, 10, 14] e propomos um método híbrido para otimização global de problemas canalizados contínuos que em uma fase heurística utiliza *simulated annealing* e em uma fase local o solver GENCAN [6]. Experimentos númericos são apresentados com o problema OVO (*Order–Value Optimization*) [4, 5] e 28 problemas clássicos com múltiplos ótimos locais [9].

2. Simulated Annealing: aspectos teóricos

Annealing é o processo utilizado para fundir um metal, onde este é aquecido a uma temperatura elevada e em seguida é resfriado lentamente, de modo que o produto final seja uma massa homogênea.

[1]apoio FAPESP
[2]ghaeser@ime.unicamp.br
[3]marcia@ime.unicamp.br

O *simulated annealing* surgiu no contexto da mecânica estatística, desenvolvido por Kirkpatrick, Gelatt e Vecchi em 1983 [10] e independentemente por Černy em 1985 [8], utilizando o algoritmo de Metropolis de 1953 [12].

Esta técnica é utilizada em problemas de otimização combinatória:

$$\begin{aligned} &\min_x f(x) \\ &\text{s.a.} \quad x \in S \end{aligned}$$

Onde $f\colon S \to \mathbb{R}$, S finito.

Neste trabalho, consideramos S como sendo uma discretização do conjunto $\{x \in \mathbb{R}^n \mid l \leq x \leq u\}$.

O algoritmo consiste em inicializar uma temperatura inicial T e um ponto inicial $i \in S$. A partir de i geramos $j \in S$ tal que j pertence a uma certa vizinhança $V(i)$ de i e aceitamos ou rejeitamos j de acordo com o critério de Metropolis. Repetimos este processo L_k vezes, de modo que um certo equilíbrio é atingido, em seguida diminuímos o valor de T e repetimos o processo. De forma esquemática temos:

```
INICIALIZAR i, T_0, L_0
k := 0
REPETIR
      PARA l = 1 ATÉ L_k
            GERAR j de V(i)
            SE f(j) ≤ f(i) ENTÃO i := j
            SENÃO SE random[0,1) < exp((f(i) − f(j))/T_k)
                  ENTÃO i := j
      k := k + 1
      CALCULAR L_k e T_k
ATÉ ‘‘critério de parada’’
```

A idéia do algoritmo é inicialmente aceitar quase todas as transições propostas, a fim de escapar de um minimizador local (para isso, T_0 deve ser suficientemente grande) e em seguida aceitar cada vez menos pontos que pioram o valor da função objetivo, sendo que no limite $T \to 0$, só aceitamos pontos que melhoram o valor da função objetivo.

Para descrevermos a teoria deste algoritmo, é preciso definir as cadeias de Markov e a proposta do algoritmo de Metropolis.

2.1. Cadeias de Markov

Definição 1. *Uma cadeia de Markov é uma seqüência de variáveis aleatórias X_k que assume valores em S tal que:*

$$\mathbf{P}\{X_k = i_k | X_{k-1} = i_{k-1}, \ldots, X_0 = i_0\} = \mathbf{P}\{X_k = i_k | X_{k-1} = i_{k-1}\},$$

além disso, a cadeia é dita finita se S é finito.

Ou seja, em uma cadeia de Markov, a probabilidade de mudarmos de um estado i para um estado j não depende dos estados anteriores.

Assim, em uma cadeia de Markov finita, podemos definir para cada k, a *matriz de transição* $P^{(k)}$, onde

$$P_{ij}^{(k)} = \mathbf{P}\{X_k = j | X_{k-1} = i\}.$$

Definição 2. *Dizemos que a cadeia de Markov é homogênea se $P^{(k)}$ não depende de k, neste caso escrevemos apenas P, para a matriz de transição. Caso contrário dizemos que a cadeia é heterogênea.*

Definição 3. *Em uma cadeia de Markov finita e homogênea, dizemos que existe a distribuição limite se, para cada $i \in S$ existe o limite*

$$q_i = \lim_{k \to +\infty} \mathbf{P}\{X_k = i | X_0 = j\},$$

além disso q_i independe de j. O vetor q é chamado de distribuição limite da cadeia de Markov.

É possível mostrar que sob certas condições, se P é a matriz de transição de uma cadeia de Markov finita e homogênea e o vetor q é uma distribuição de probabilidade em S e $q_i P_{ij} = q_j P_{ji}$ $\forall\, i, j \in S$, então q é a distribuição limite desta cadeia de Markov. Para detalhes veja [9].

2.2. Algoritmo de Metropolis

Uma questão central que precisa ser respondida é a seguinte: "dado um conjunto finito S e uma distribuição de probabilidade q em S, como construir uma cadeia de Markov em S com distribuição limite q?" O algoritmo de Metropolis, responde a esta pergunta.

A seguir definimos o algoritmo.

Seja $G \in \mathbb{R}^{n \times n}$ uma matriz simétrica, onde $n < +\infty$ é a cardinalidade de S e cada linha de G é uma distribuição de probabilidade. A matriz G é chamada *matriz de geração* e G_{ij} é a probabilidade de gerar j a partir de i.

Seja $\alpha_{ij} = \min\left\{\dfrac{q_j}{q_i}, 1\right\}$, a *matriz de aceitação* α. α_{ij} é chamado de *critério de Metropolis*.

O algoritmo consiste em: dado $X_k = i$, gerar j a partir de i de acordo com as probabilidades G_{it}, $\quad \forall\, t \in S$. Aceitar $X_{k+1} = j$ com probabilidade α_{ij}, ou rejeitar j, isto é, $X_{k+1} = i$ com probabilidade $1 - \alpha_{ij}$.

Deste modo X_k é uma cadeia de Markov finita e homogênea com

$$P_{ij} = \alpha_{ij} G_{ij}, \text{ se } i \neq j \text{ e } P_{ii} = 1 - \sum_{j \in S, j \neq i} P_{ij}.$$

Note que no algoritmo de Metropolis não é necessário conhecermos o valor da constante normalizadora da distribuição q, pois o algoritmo só depende de q na razão $\dfrac{q_j}{q_i}$.

Usando a simetria de G, temos que, para $i \neq j$:

$$\begin{aligned} q_i P_{ij} &= q_i \alpha_{ij} G_{ij} \\ &= q_i \min\left\{\frac{q_j}{q_i}, 1\right\} G_{ij} \\ &= \begin{cases} q_j G_{ij} & \text{se } q_j \leq q_i \\ q_i G_{ij} & \text{se } q_j > q_i \end{cases} \\ &= \begin{cases} q_i G_{ji} & \text{se } q_i < q_j \\ q_j G_{ji} & \text{se } q_i \geq q_j \end{cases} \\ &= q_j \min\left\{\frac{q_i}{q_j}, 1\right\} G_{ji} \\ &= q_j \alpha_{ji} G_{ji} \\ &= q_j P_{ji}. \end{aligned}$$

Assim, é possível mostrar que a cadeia de Markov construída pelo algoritmo de Metropolis possui distribuição limite q, se q não é uniforme em S e $q_i > 0 \quad \forall\, i \in S$.

O *simulated annealing* consiste na aplicação do algoritmo de Metropolis repetidas vezes, onde a cada execução a distribuição limite desejada é modificada.

2.3. Distribuição de Boltzmann

Dado $f : S \to \mathbb{R}$, S finito, a distribuição de Boltzmann é definida por:

$$q_i(T) = \frac{1}{N_0(T)} \exp\left(-\frac{f(i)}{T}\right) \quad \forall\, i \in S,$$

onde $N_0(T)$ é uma constante normalizadora.

A escolha desta distribuição se justifica pela propriedade:

$$\lim_{T \to 0^+} q_i(T) = q_i^* \stackrel{\text{def}}{=} \begin{cases} \frac{1}{|S_{\texttt{opt}}|} & \text{se } i \in S_{\texttt{opt}} \\ 0 & \text{caso contrário} \end{cases},$$

onde $S_{\texttt{opt}} = \{i \in S | f(i) \leq f(j), \quad \forall\, j \in S\}$ e $|S_{\texttt{opt}}|$ é a cardinalidade de $S_{\texttt{opt}}$.

Isto é, conforme $T \to 0$ a distribuição de Boltzmann tende para uma distribuição uniforme nos minimizadores globais.

Note que $q_i(T) > 0 \quad \forall i \in S$ e $\forall T > 0$, assim obtemos o seguinte resultado:

$$\lim_{T \to 0^+} \lim_{k \to +\infty} \mathbf{P}_T\{X_k = i\} = \lim_{t \to 0^+} q_i(T) = q_i^*,$$

ou seja,

$$\lim_{T\to 0^+}\lim_{k\to+\infty}\mathbf{P}_T\{X_k\in S_{\texttt{opt}}\}=1.$$

Note que este resultado também se verifica no caso particular em que a distribuição de Boltzmann é a distribuição uniforme em S, pois neste caso f é constante em S e o resultado é trivial.

A idéia do *simulated annealing* é obter $i\in S$ com distribuição q^*. Para isso construímos a cadeia de Markov homogênea com T fixo até atingirmos a distribuição limite, em seguida diminuímos T e restauramos o equilíbrio atingindo novamente a distribuição limite para a nova temperatura, repetimos o processo até $T\approx 0$.

Em [1] é obtido um resultado de convergência como este, para um algoritmo onde o número de iterações para cada T fixo é finito, com a hipótese adicional de que o decréscimo da temperatura deve ser suficientemente lento.

Surgem então alguns parâmetros em aberto quanto à implementação do algoritmo:

- o valor inicial para a temperatura (T_0);
- uma função para ditar o decréscimo da temperatura;
- um valor final para a temperatura, ou seja, um critério de parada;
- um número finito de transições para cada temperatura (L_k).

Para implementarmos o *simulated annealing* devemos ter um *esquema de resfriamento*, que especifica os parâmetros acima.

3. Simulated Annealing com acelerador local

No algoritmo que propomos (SA–GENCAN) para minimização em caixas com função objetivo diferenciável, o *simulated annealing* é aplicado, sem discretização explícita do domínio, sendo assim, um novo elemento deve ser escolhido da vizinhança do ponto atual, sendo esta vizinhança um subconjunto de $\mathbb{R}^n$ com interior em geral não vazio.

Fixado $\varepsilon>0$, definimos a vizinhança V de cada ponto $\bar{x}$ da caixa, como o conjunto dos pontos da caixa que distam no máximo ε de $\bar{x}$, na norma infinito. Um novo ponto é obtido gerando uma variável aleatória uniforme na vizinhança, o que pode ser feito por componentes já que cada vizinhança também é uma caixa, devido à utilização da norma infinito.

A cada troca de temperatura, empregamos o solver GENCAN para problemas diferenciáveis, não lineares com restrições de caixa, proposto e implementado por Birgin e Martínez em 2002, [6]. Trata-se de um método de restrições ativas com gradiente espectral projetado, onde a cada iteração uma aproximação quadrática do problema é resolvido em uma região de confiança. Em [6] os autores demonstram a convergência global do método.

A solução encontrada pelo solver local não é empregada como ponto inicial para o próximo valor de temperatura, ao invés disto, construímos uma lista de

soluções encontradas por GENCAN, sendo que as iterações seguem com o ponto que possuíamos antes de aplicarmos GENCAN. Acreditamos que deste modo estamos sendo mais coerentes com a teoria de *simulated annealing*, pois evitamos mudanças bruscas no valor da função objetivo ao longo da cadeia de Markov gerada.

Executamos GENCAN com os parâmetros padrões, e acrescentamos um novo critério de parada: retornamos o ponto atual do GENCAN como solução se ele estiver à uma distância euclidiana menor do que 10^{-2} de algum dos pontos da lista. Se este critério não for acionado, acrescentamos este novo ponto à lista (mesmo que o GENCAN não tenha declarado convergência para um minimizador local). O objetivo da lista, além de armazenar a melhor solução, é evitar que GENCAN encontre uma solução já encontrada anteriormente.

O esquema de resfriamento implementado para o *simulated annealing* foi o descrito em [13], com os valores típicos para os parâmetros, juntamente com o critério de parada descrito em [1, 7]. Além disto limitamos em 100 o número máximo de trocas de temperatura.

Segue abaixo um esboço do algoritmo:

INICIALIZAR x, T_0, L_0
$k := 0$
REPETIR
  PARA $l = 1$ ATÉ L_k
    GERAR y de $V(x)$
    SE $f(y) \leq f(x)$ ENTÃO $x := y$
    SENÃO SE random$[0,1) < \exp\left(\dfrac{f(x) - f(y)}{T_k}\right)$
      ENTÃO $x := y$
  $k := k + 1$
  CALCULAR L_k e T_k
  EXECUTAR GENCAN A PARTIR de x
ATÉ ‘‘critério de parada’’

4. Experimentos numéricos

Resolvemos o problema OVO e 28 problemas clássicos da literatura, de pequeno porte e múltiplos ótimos locais [9], por *simulated annealing* puro e SA–GENCAN. A implementação foi feita em Fortran 77 em uma máquina Pentium 4, 3.5 GHz, 2Gb Ram.

A seguir definimos o problema OVO (*Order–Value Optimization*), de acordo com [2, 3, 4, 5].

Dadas m funções $f_1, \ldots, f_m$ definidas no conjunto $\Omega \subseteq \mathbb{R}^n$ e um inteiro positivo $p \leq m$, definimos as m funções índices $i_1, \ldots, i_m : \Omega \to \{1, 2, \ldots, m\}$ de modo que $\{i_1(x), i_2(x), \ldots, i_m(x)\}$ é uma permutação de $\{1, 2, \ldots, m\}$ e

$$f_{i_1(x)}(x) \leq f_{i_2(x)}(x) \leq \cdots \leq f_{i_p(x)}(x) \leq \cdots \leq f_{i_m(x)}(x) \quad \forall\, x \in \Omega.$$

Assim, podemos definir o problema OVO:

$$\min_x \sum_{j=1}^{p} f_{i_j(x)}(x)$$
$$\text{s.a.} \quad x \in \Omega$$

Em particular, queremos ajustar o modelo $y(t,x) = x_1 + x_2 t + x_3 t^2 + x_4 t^3$ em um conjunto de dados (t_i, y_i), $i = 1, \ldots, m$, desprezando os $m - p$ piores ajustes. As funções f_i são as funções erro $f_i(x) = (y(t_i, x) - y_i)^2$.

Sendo $\Omega = [-3, 3]^4$ e dada a solução $x^* = (x_1^*, x_2^*, x_3^*, x_4^*)$ escolhida aleatoriamente em Ω, $m = 101$ e $p = 80$ definimos $t_i = -1 + 0.02(i-1)$ e $w_i = y(t_i, x^*) \quad \forall\, i = 1, 2, \ldots\ m$.

Escolhemos aleatoriamente $m - p = 21$ elementos de $\{1, 2 \ldots, 101\}$, a saber $p_1, p_2, \ldots, p_{21}$ e definimos

$$y_i = \begin{cases} 10 & \text{se} \quad i \in \{p_1, p_2, \ldots, p_{21}\} \\ w_i & \text{c.c.} \end{cases}$$

A solução deste problema é x^* com valor nulo para a função objetivo.

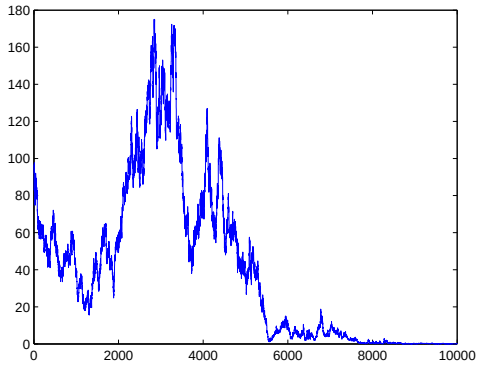

Figura 1: Valor da função objetivo de cada elemento da cadeia de Markov gerada.

Pela Figura 1 percebemos que de fato, nas iterações iniciais o método oscila entre pontos bons e ruins, enquanto que nas iterações finais, pontos com valores piores da função objetivo não são aceitos.

Na Figura 2 temos a cadeia de Markov gerada como no exemplo anterior, com a diferença de que antes de cada troca de temperatura melhoramos o valor da função objetivo através da execução de GENCAN, cada execução está sinalizada na figura com o símbolo *. Notamos neste caso uma oscilação pequena no valor da função objetivo após gerar 3000 pontos, o que demora mais para ocorrer no caso em que apenas *simulated annealing* é executado.

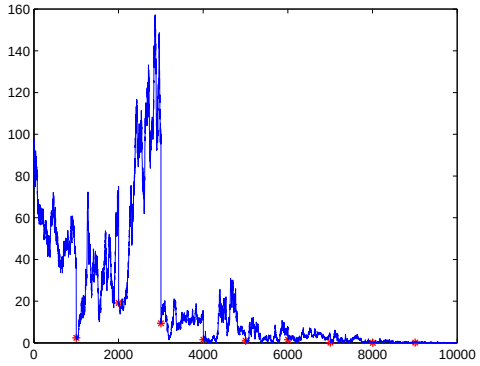

Figura 2: Valor da função objetivo da cadeia de Markov gerada por SA–GENCAN

A Tabela 1 mostra o desempenho de SA–GENCAN, GENCAN e *simulated annealing* em 28 problemas de pequeno porte e com múltiplos ótimos locais. Cada problema foi executado 20 vezes, com diferentes aproximações iniciais, sendo que o problema é considerado resolvido se o desvio percentual em relação ao ótimo global é menor que 0.01.

Tabela 1: Comparação entre os algoritmos

	problemas resolvidos	avaliações de função	avaliações de gradiente
SA–GENCAN	98.92%	10378	1593
GENCAN	57.32%	183	62
simulated annealing	69.28%	9619	-

A Tabela 2 mostra a freqüência com que cada critério de parada foi acionado em SA–GENCAN.

Tabela 2: Critérios de parada acionados em SA–GENCAN

Critérios de Parada	Porcentagem
Variação pequena da função objetivo	35.96%
Número máximo de iterações	47.31%
Número máximo de avaliações de função	3.28%
Temperatura muito pequena	3.85%
Nenhuma melhora nas últimas iterações	9.61%

5. Conclusões

Observamos que apenas GENCAN para a obtenção do ótimo global não é eficiente, uma vez que o número de problemas resolvidos é bem reduzido, o que era de se esperar pois GENCAN é um método local e os problemas apresentados possuem diversos minimizadores locais. O que pode ser feito para melhorar sua performance é executar GENCAN mais de uma vez com aproximações iniciais escolhidas aleatoriamente no espaço de busca, neste caso os resultados são semelhantes aos obtidos por SA–GENCAN nos problemas de dimensão pequena e se torna ineficiente para problemas de dimensões maiores. Além disto, em SA–GENCAN o procedimento realizado pelo *simulated annealing* escolhe aproximações iniciais mais promissoras o que resulta numa vantagem para este algoritmo.

Comparamos o desempenho de SA–GENCAN com o desempenho de um acelerador local para *simulated annealing* sem derivadas, a saber o método DBMS [11]. Percebemos que ao utilizar um solver local robusto como GENCAN, ao invés de uma busca local sem derivadas, temos uma forte melhora de performance.

Analisando os resultados temos que SA–GENCAN é o que consegue resolver o maior número de problemas, embora o custo computacional seja substancialmente maior, como era de se esperar já que em SA–GENCAN são feitas chamadas de GENCAN e também há o mesmo esforço presente em *simulated annealing*.

Maiores detalhes teóricos, sobre os testes númericos e uma tentativa de aplicar a mesma idéia em problemas com restrições pode ser encontrada em [9].

Abstract. In this work we present the theory behind simulated annealing and propose a hybrid method for continuous box–constrained global optimization. On the heuristic phase simulated annealing is used and GENCAN is used on the local phase.

Referências

[1] E.H.L. Aarts and J.H.M. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons, 1989.

[2] R. Andreani, C. Dunder, and J.M. Martínez. Order-Value Optimization: formulation and solution by means of a primal Cauchy method. *Mathematical Methods of Operations Research*, 58:387–399, 2003.

[3] R. Andreani, C. Dunder, and J.M. Martínez. Nonlinear-programming reformulation of the Order-Value Optimization problem. *Mathematical Methods of Operations Research*, 61:365–384, 2005.

[4] R. Andreani, J.M. Martínez, M. Salvatierra, and F. Yano. Global Order–Value Optimization by means of a multistart harmonic oscillator tunneling strategy. *Global Optimization: from Theory to Implementation*.

[5] R. Andreani, J.M. Martínez, M. Salvatierra, and F. Yano. Quasi-Newton methods for Order–Value Optimization and Value-at-Risk calculations. *Pacific Journal of Optimization*, 2:11–33, 2006.

[6] E.G. Birgin and J.M. Martínez. Large–scale active–set box–constrained optimization method with spectral projected gradients. *Computational Optimization and Applications*, 23:101–125, 2002.

[7] M. F. Cardoso, R. L. Salcedo, and S. Feyo de Azevedo. The simplex–simulated annealing approach to continuous non–linear optimization. *Computers Chem. Engng.*, 20(9):1065–1080, 1995.

[8] V. Černy. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985.

[9] G. Haeser. Algoritmo duas fases em otimização global. *Tese de Mestrado, T/UNICAMP H119a*, 2006.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[11] J. Ma, P. Tian, and D.-M. Zhang. Global optimization by Darwin and Boltzmann mixed strategy. *Computers & Operations Research*, 27:143–159, 2000.

[12] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[13] P. Siarry, G. Berthiau, F. Durbin, and J. Haussy. Enhanced simulated annealing for globally minimizing functions of many continuous variables. *ACM Transactions on Mathematical Software*, 23(2):209–228, 1997.

[14] M. W. Trosset. What is simulated annealing? *Optimization and Engineering*, 2(2):201–213, 2001.