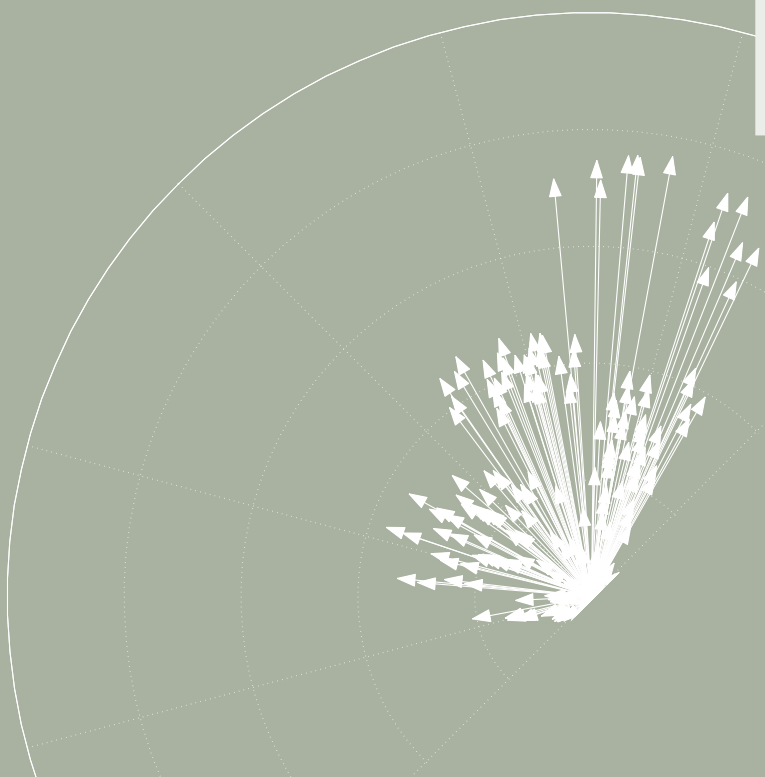
MANUAL DE COMPUTAÇÃO EVOLUTIVA E METAHEURÍSTICA

ANTÓNIO GASPAR-CUNHA
RICARDO TAKAHASHI
CARLOS HENGGELER ANTUNES
COORDENADORES



IMPRENSA DA UNIVERSIDADE DE COIMBRA
COIMBRA UNIVERSITY PRESS

EDITORA ufmg

CAPÍTULO 8

Recozimento Simulado

Marconi A. Pereira *João A. Vasconcelos*

Departamento de Engenharia Elétrica
Universidade Federal de Minas Gerais

Recozimento Simulado[1] é uma metaheurística inspirada no processo físico de recozimento de um sólido para obtenção de estados de baixa energia na área da física da matéria condensada. Esse processo consiste em aquecer o sólido até atingir sua temperatura de fusão, para que a matéria passe do estado sólido para o líquido; posteriormente, a temperatura deve ser lentamente diminuída para evitar estados meta-estáveis. O objetivo deste processo é obter a matéria no estado cristalino, ou seja, com energia mínima.

No estado líquido a matéria possui grande quantidade de energia e suas partículas são dispostas aleatoriamente. Em contrapartida, no estado sólido cristalizado as partículas são dispostas de forma extremamente estruturada, com mínima energia. É muito importante que o resfriamento para a passagem do estado líquido para o sólido ocorra de maneira lenta e cuidadosa, pois do contrário, a matéria pode parar em um estado sólido intermediário e não mais se cristalizar. A Figura 8.1 mostra um esquema do recozimento.

O conceito de recozimento (annealing) foi introduzido em otimização combinatória na década de 80 por Kirkpatrick et al. (1982) e Kirkpatrick (1984). Assim como o recozimento procura levar a matéria

[1] Do inglês *simulated annealing*. No Brasil essa expressão vem sendo traduzida como *recozimento simulado*, guardando o significado original da expressão em língua inglesa, que faz referência ao processo metalúrgico de *recozimento* de metais. Em Portugal, a expressão vem sendo traduzida como *arrefecimento simulado*.

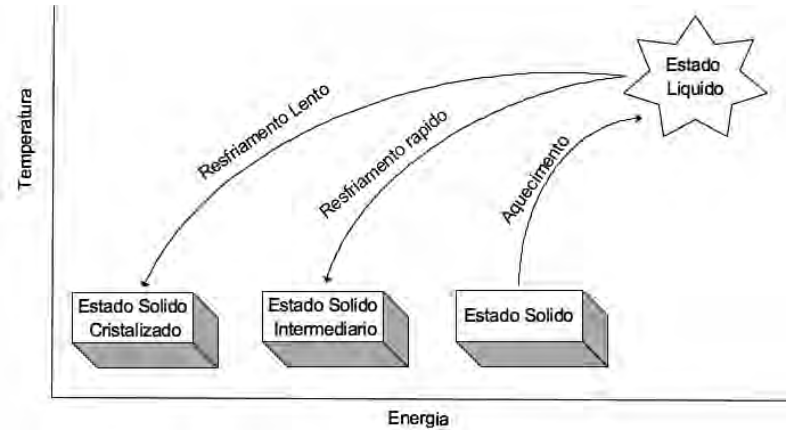


Figura 8.1: Recozimento. A matéria é aquecida (bloco à direita) e depois resfriada lentamente (bloco à esquerda). Um resfriamento rápido leva a matéria para um estado sólido de energia intermediária (bloco central).

para um estado de energia mínima, a otimização consiste em encontrar uma solução que minimize uma determinada função objetivo. O problema de otimização pode ser escrito de forma genérica como:

$$\begin{aligned} &\text{minimize } \overline{f}(\overline{x}) \\ &\text{sujeito a: } \begin{cases} \overline{g}(\overline{x}) \leq \overline{0} \\ \overline{h}(\overline{x}) = \overline{0} \end{cases} \end{aligned} \tag{8.1}$$

onde $\overline{x} \in \mathbb{R}^n$, $\overline{f}\colon \mathbb{R}^n \to \mathbb{R}^m$, $\overline{g}\colon \mathbb{R}^n \to \mathbb{R}^k$, $\overline{h}\colon \mathbb{R}^n \to \mathbb{R}^l$, com n, m, k e $l \in \mathbb{N}$. Se $n = 1$, o problema é mono-objetivo. Caso contrário, ele é multiobjetivo.

Na década de 50, Metropolis publicou um trabalho (Metropolis et al. (1953)) no qual o processo físico de recozimento foi modelado em um algoritmo que simula o resfriamento de um sólido em direção ao equilíbrio térmico. Esse algoritmo é baseado em técnicas de Monte Carlo (Gilks et al. (1995)), onde é gerada uma sequência de estados do sólido obtidos da seguinte maneira:

- Aplica-se uma perturbação em um estado corrente i do sólido cuja energia atual é E_i, para gerar um novo estado j. A energia desse novo estado é E_j.
- Se a diferença de energia $E_j - E_i$ for menor ou igual a zero, o estado j é aceito.
- Se a diferença é maior que zero, o estado j é aceito de acordo com a seguinte probabilidade:

$$p = \exp\left(\frac{E_i - E_j}{k_b T}\right), \tag{8.2}$$

onde T denota a temperatura e k_b é uma constante física chamada Constante de Boltzmann. Esse critério de aceitação do estado j é conhecido como critério de Metropolis.

A temperatura deve diminuir lentamente para que o equilíbrio térmico seja alcançado. O algoritmo de Metropolis deve gerar, portanto, um grande número de transições a fim de atingir esse equilíbrio. As transições geradas caracterizam uma distribuição de Boltzmann (Aarts e Korst (1989)). Analisando a equação 8.2 percebe-se que a probabilidade de aceite do estado é alta quando o valor da temperatura é alto. À medida em que a temperatura diminui, a probabilidade de aceite do novo estado também

Processo de Recozimento	Recozimento Simulado
Aplicado na busca pelo Equilíbrio Térmico da Matéria	Visa a Otimização de uma função
Variação energética	Variação da função objetivo
Procura determinar o estado de mínima energia	Procura determinar o valor mínimo da função objetivo
Baseado na diminuição da temperatura (T)	Baseado na diminuição do parâmetro de controle (c)
Pode parar em estados intermediários de energia da matéria, não atingindo o estado de mínima energia	Pode parar em mínimos locais da função objetivo, não atingindo o mínimo global

Tabela 8.1: Analogia entre o processo de recozimento e o algoritmo de recozimento simulado

diminui. De acordo com Aarts e Korst (1989), a probabilidade do sólido estar no estado i com energia E_i e temperatura T é dada por:

$$P_T\{X = i\} = \frac{1}{Z(T)} \exp\left(\frac{-E_i}{k_b T}\right), \tag{8.3}$$

onde X é uma variável estocástica que indica o estado corrente do sólido e $Z(T)$ é a função de partição definida por:

$$Z(T) = \sum_j \exp\left(\frac{-E_i}{k_b T}\right), \tag{8.4}$$

onde o somatório cobre todos os estados possíveis.

Conforme informado anteriormente, o algoritmo de Metropolis é a base do recozimento simulado. A implementação do recozimento simulado será detalhada na seção seguinte.

1. Implementação do Recozimento Simulado

O recozimento simulado utiliza o algoritmo de Metropolis para simular o equilíbrio térmico. Assim, pode-se assumir uma analogia entre o processo físico e o processo de otimização combinatória baseada nas seguintes equivalências (Aarts e Korst (1989)):

- Soluções intermediárias de um problema de otimização combinatória equivalem às etapas de resfriamento da matéria;
- O valor da função objetivo do problema de otimização equivale à energia do sólido no processo físico de resfriamento em direção ao equilíbrio térmico.

A Tabela 8.1 mostra uma analogia entre o processo de recozimento e a aplicação do recozimento simulado em problemas de otimização.

O recozimento simulado assume a existência de uma vizinhança, que pode ser encontrada pelo mecanismo de busca. Assim, são utilizadas as seguintes definições (Aarts e Korst (1989)):

Definição 8.1 - (S, f) *denota uma amostra de um problema de otimização e* i *e* j *duas soluções com custo* $f(i)$ *e* $f(j)$*, respectivamente. Assim, o critério de aceite determina quando* j *é aceito em substituição a* i *pela aplicação da probabilidade de aceite descrita na equação 8.5.*

$$P_c\{\text{aceite}j\} = \begin{cases} 1 & \text{se } f(j) \leq f(i) \\ \exp\left(\frac{f(i)-f(j)}{c}\right) & \text{se } f(j) > f(i) \end{cases} \tag{8.5}$$

onde $c \in \mathbb{R}^+$ denota o parâmetro de controle.

Definição 8.2 - *Uma transição é uma ação que resulta na transformação de uma solução corrente em uma solução subsequente. Essa ação consiste nas seguintes etapas: (a) aplicação de uma perturbação, ou seja, mecanismo de geração de um novo estado; (b) aplicação do critério de aceitação.*

Seja c_k o valor do parâmetro de controle e L_k o número de transições geradas na k-ésima iteração do Algoritmo de Metropolis. Com estas definições, o pseudocódigo do SA pode ser apresentado (Algoritmo 1).

Algoritmo 1 Pseudocodigo para o algoritmo de recozimento simulado

```
1: INICIALIZA(i_inicial, c_0, L_0);
2: k ← 0;
3: i ← i_inicial;
4: enquanto não se atinge o critério de parada faça
5:    para l ← 1 to L_K faça
6:       GERE (j à partir de S_i);
7:       se f(j) ≤ f(i) então
8:          i ← j;
9:       senão
10:         se exp((f(i)−f(j))/c_k) > random[0,1) então
11:            i ← j;
12:         fim se
13:      fim se
14:   fim para
15:   k ← k + 1;
16:   CALCULA_TAMANHO(L_k);
17:   CALCULA_CONTROLE(c_k);
18: fim enquanto
```

Além de aceitar redução (minimização) na função de custo (linhas 7 e 8), o recozimento simulado pode aceitar uma deterioração (aumento) na função, com uma certa probabilidade (definição 1 implementada nas linhas 10 e 11 do algoritmo 1). Nas primeiras iterações do algoritmo, essa aceitação será maior, pois os valores de c_k serão maiores. À medida em que o algoritmo prossegue, os valores de c_k irão diminuindo (linha 17), fazendo com que a probabilidade de aceitação de deteriorações na função de custo também diminua. O fato do algoritmo aceitar algumas deteriorações na função de custo faz com que ele escape de mínimos locais.

O recozimento simulado utiliza alguns parâmetros que podem influir diretamente na eficiência do algoritmo: Valor inicial do parâmetro de controle (c_k), taxa de diminuição deste valor e o número de ciclos de temperatura (L_k) sobre cada valor de temperatura. Estes parâmetros precisam ser escolhidos corretamente para cada tipo de problema, pois podem ser determinantes para se obter um ótimo global. Esses valores devem ser escolhidos de acordo com o problema o qual se deseja otimizar. Versões mais recentes do algoritmo proporcionam formas de se escolher bons valores para esses parâmetros.

O valor de c_k, geralmente, é decrementado da seguinte maneira:

$$c_{k+1} = \alpha.c_k, k = 1, 2, \ldots \tag{8.6}$$

onde α é uma constante menor que 1, geralmente variando entre 0.8 e 0.99.

A quantidade de perturbações (L_k), em cada valor do parâmetro de controle (c_k), pode estar relacionada a uma combinação de diversos fatores, tais como o número máximo de iterações, taxa mínima de queda de energia, entre outros. O importante é que se atinja o equilíbrio térmico para cada valor de c_k.

2. Aplicações

O algoritmo de recozimento simulado é aplicado com grande sucesso em problemas de otimização combinatória: Problemas discretos, como por exemplo, o problema do caixeiro viajante e projeto de equipamentos eletrônicos, além de problemas contínuos como problemas de projeto das formas geométricas de equipamentos, são resolvidos com recozimento simulado. A seguir, será descrito como estes problemas podem ser modelados.

Problemas Discretos

O problema do caixeiro viajante consiste em encontrar o menor caminho que percorre todas as cidades de um dado conjunto. Formalmente pode-se definir esse problema da seguinte maneira:

Definição 8.3 - *Problema do Caixeiro Viajante: Seja $C = c_1, \ldots, c_n$ um conjunto de n cidades e $D = [d_{ij}]$ a matriz de adjacência cujo elemento d_{ij} denota a distância entre a cidade i e a cidade j. O problema consiste em encontrar o menor caminho no qual todas as cidades são visitadas.*

O problema do caixeiro viajante pode ser resolvido da seguinte maneira, utilizando o algoritmo recozimento simulado (Aarts e Korst (1989)):

- O espaço de busca das soluções S corresponde ao conjunto que contem todas as possíveis permutações entre caminhos que ligam todas as cidades. Cada caminho é dado por $\pi = (\pi(1), \ldots, \pi(n))$, onde $\pi(i), i = 1, \ldots, n$ denota a cidade sucessora da cidade i. A cidade número 1 é a sucessora da n-ésima cidade.
- Gera-se um estado inicial ($i_{inicial}$) que corresponde a um caminho entre as cidades, gerado aleatoriamente; considere $c_0 = 0.10 * n$, onde n é o número total de cidade e $L_k = 10 * n$;
- A função de custo a ser minimizada é dada por :

$$f(\pi) = \sum_{i=1}^{n} d_{i,\pi(i)} \tag{8.7}$$

- A perturbação consiste em gerar novas soluções (caminhos) escolhendo aleatoriamente duas cidades e invertendo suas posições na sequência de cidades a serem visitadas;
- A diferença em custo entre duas cidades p e q, que permutaram suas posições, pode ser calculada de maneira incremental a partir da seguinte expressão:

$$\Delta f = -a - b - c - d + a' + b' + c' + d' \tag{8.8}$$

onde a e b são as distâncias entre a cidade p e sua antecessora e sucessora respectivamente, c e d são as distâncias entre a cidade q e sua antecessora e sucessora; a' e b' são as distâncias entre a cidade q e sua nova antecessora e nova sucessora respectivamente e c' e d' são as distâncias entre a cidade p e sua nova antecessora e nova sucessora. A Figura 8.2 mostra um exemplo de permutação de cidades.

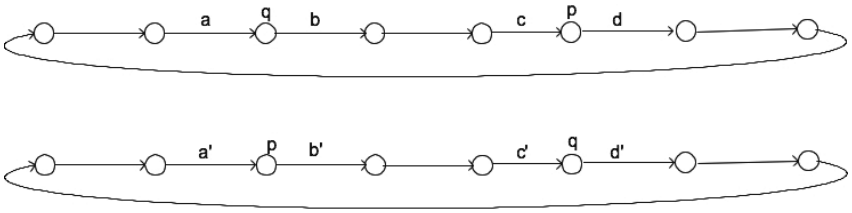


Figura 8.2: Problema do Caxeiro Viajante. O SA permuta o caminho entre duas cidades a fim de procurar a menor rota.

Um outro exemplo de aplicação citado é o problema de projeto de equipamentos. Trata-se de um problema bastante complexo, que na prática pode ter de 10^3 a 10^5 variáveis, além de vários objetivos conflitantes a serem otimizados. Um exemplo simples deste tipo de problema é mostrado em (Rutenbar (1989)): Deseja-se projetar a disposição dos componentes de um circuito integrado com o objetivo de otimizar o cabeamento (*wirability*) entre estes. O circuito é modelado como uma malha retangular, onde cada módulo deve estar em um dos nós da malha retangular e deve ter no máximo quatro terminais elétricos. Um conjunto de módulos é interligado, compondo uma rede, e posteriormente essas redes são interligadas de modo a compor o circuito como um todo. Devem-se posicionar os módulos das redes para otimizar o cabeamento.

A modelagem do problema é feita da seguinte forma:

- Identifica-se o modelo que define o espaço de busca das soluções, ou seja, as soluções viáveis. Em suma, deve-se identificar as restrições do problema;
- A perturbação em uma dada configuração i consiste em realizar uma alteração possível (que atenda às restrições do problema) da posição de um módulo da rede. O mais comum seria a troca de posição entre dois módulos. Para cada temperatura são executadas $100M$ perturbações, onde M é o número de módulos do circuito;
- A função de custo consiste no cálculo do comprimento total dos fios necessários para interligar o circuito. Em situações reais, outras variáveis poderiam ser consideradas, como por exemplo, o congestionamento do cabeamento (*wiring congestion*);
- O critério de parada do processo consiste na identificação de três quedas sucessivas de temperatura sem que a função de custo decaia em mais de 1%.

A Figura 8.3 mostra a conexão lógica dos módulos que compõem o circuito integrado. A Figura 8.4 mostra uma possível configuração (design) que implementa o circuito em questão.

Considerando a execução do problema de design do circuito integrado conforme descrito anteriormente, utilizando aproximadamente 800 módulos, obtém-se um gráfico de custo em função da temperatura (parâmetro de controle) conforme disposto na Figura 8.5.

Ao longo do processo, a temperatura é diminuída (da direita para a esquerda) ao mesmo tempo que o custo da função objetivo também diminui. Em alguns pontos há um pequeno acréscimo na função custo, mas posteriormente a função volta a cair. Esse comportamento contribui para que o algoritmo saia de mínimos locais da função objetivo.

Como o recozimento simulado é um processo estocástico, o resultado final de uma execução pode ser diferente do resultado obtido em outra. A Figura 8.6 mostra os resultados finais obtidos em 20 execuções do recozimento simulado. A figura mostra ainda uma linha informando a média e a variância (σ) dos valores finais.

Um exemplo real de concepção da planta baixa (*floorplannig*) de um chip é detalhado em (Rutenbar (1989)).

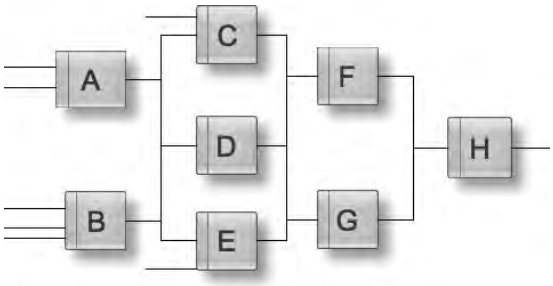


Figura 8.3: Conexão Lógica dos Módulos (Rutenbar (1989)).

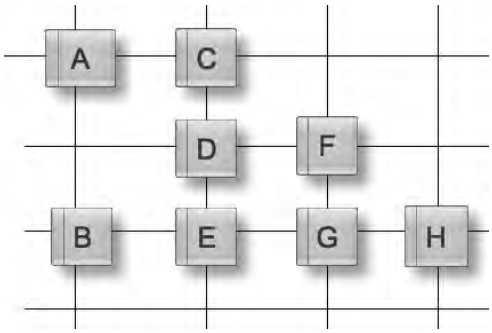


Figura 8.4: Configuração física do circuito na forma de grid (Rutenbar (1989)).

Problemas Contínuos

O recozimento simulado também é aplicado com sucesso em problemas cujas variáveis são contínuas. Uma das primeiras formas de se abordar este tipo de problema foi apresentada em Corana et al. (1987). Posteriormente, Vasconcelos et al. (1996) apresentaram uma melhora no primeiro algoritmo, acrescentando estratégias de busca tabu. Este capítulo irá detalhar essa última versão do algortimo, por se tratar de uma versão mais robusta e eficiente.

Seja $x \in \mathbb{R}^n$ um vetor com os seguintes componentes $x_1, \ldots, x_n$. Seja $f(x)$ a função a ser minimizada, onde $a_1 < x_1 < b_1, \ldots, a_n < x_n < b_n$ são suas n variáveis com os respectivos intervalos contínuos e finitos. A função $f(x)$ não precisa necessariamente ser continua, porém possui limites. O algoritmo apresentado por Vasconcelos et al. (1996) é descrito no Algoritmo 2.

Os parâmetros e variáveis utilizados no algoritmo são detalhados na tabela 8.2.

A implementação do recozimento simulado para problemas com variáveis contínuas é muito semelhante à versão para problemas com variáveis discretas. A diferença está basicamente em duas etapas:

- Atualização dos pontos. A versão para problemas de variáveis contínuas utiliza, além do ponto anterior(x_i) e de um valor randômico (dentro do intervalo $[-1, 1]$), mais um elemento: a i-ésima dimensão do vetor H.

- Atualização do vetor de direções. O parâmetro C_i controla a variação na direção u. Resultados práticos mostram que um bom valor para esse parâmetro é $C_i = 0.2, i = 1, 2, \ldots, n$.

No trabalho apresentado por Vasconcelos et al. (1996), o recozimento simulado foi implementado a fim de minimizar as funções Exponencial (8.9) e Rastrigin (8.10). Essas funções, dentre outras,

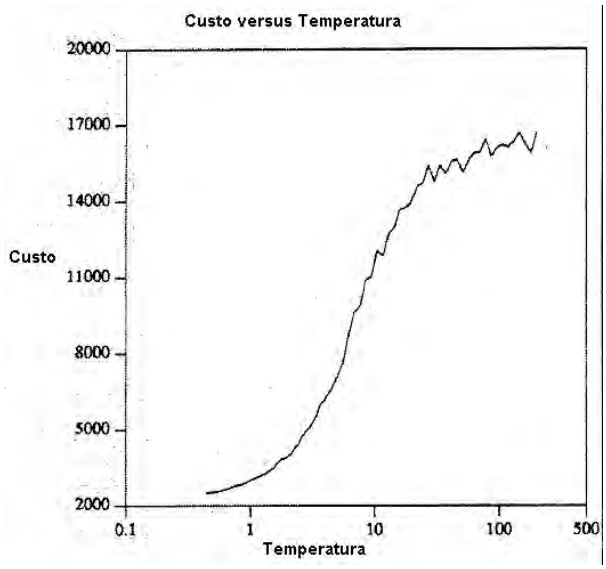


Figura 8.5: Gráfico do custo versus temperatura (Rutenbar (1989)).

são muito utilizadas em testes de algoritmos de otimização com variáveis discretas, uma vez que apresentam diversos mínimos locais. Um bom algoritmo de otimização deve ser capaz de escapar desses mínimos locais de forma a se aproximar do valor mínimo global. A função Exponencial foi definida como:

$$f(x) = 20 - e^{-\sum(x_i-1.5)^2+1} + e^{-\sum(x_i-2.5)^2+1.05} - e^{-\sum(x_i-3.5)^2+1.1} \tag{8.9}$$

onde $0 \leq x_i \leq 10 \quad i = 1, 2.$

Essa função possui dois mínimos locais. O mínimo global é o ponto $x^* = 3.59585; 3.59585)^T$. O valor da função nesso ponto é $f(x^*) = 17.30889$.

A função Rastrigin é dada por:

$$f(x) = 10n + \sum_{i=1}^{n}[(x_i)^2 - 10\cos(2\pi)(x_i)] \tag{8.10}$$

onde $n \in \mathbb{N}$ é o número de variáveis, $\overline{x} \in \mathbb{R}^n$ e $x_i \in [-5.12; 5.12] \quad \forall i = 1, 2, \ldots, n.$

A função Rastrigin é continua e multimodal. Para $x_i \in [-5.12; 5.12]$ e $n = 10$, existem 10^n minimos locais e um mínimo global em $x^* = (0; \ldots; 0)^T$ onde $f(x^*) = 0$.

A figura 8.7 mostra um gráfico das funções Exponencial e Rastrigin.

O experimento apresentado em Vasconcelos et al. (1996) utilizou como critério de parada o alcance de um valor mínimo da temperatura. Os resultados práticos mostraram que o algoritmo apresentado obteve melhores resultados que a implementação clássica do recozimento simulado.

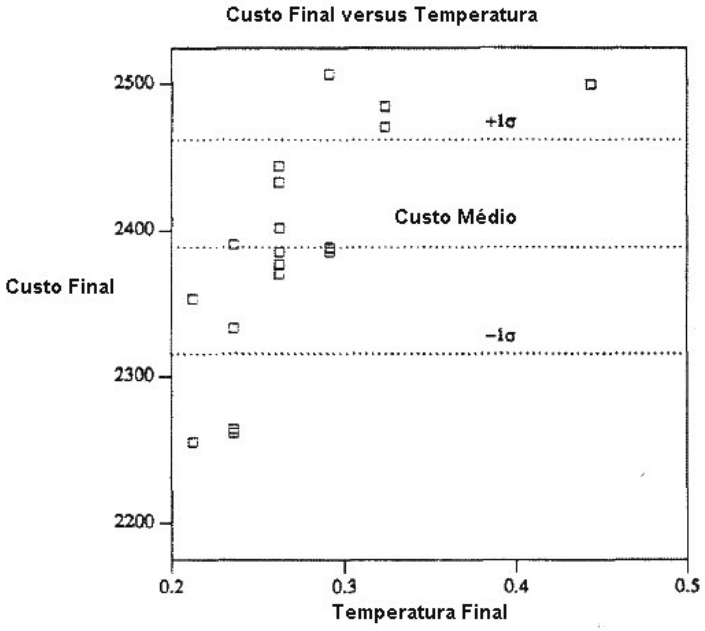


Figura 8.6: Resultados obtidos em 20 execuções do recozimento simulado para o projeto do circuito integrado (Rutenbar (1989)).

3. Recozimento Simulado Multiobjetivo

A grande maioria dos problemas reais, possui muitos objetivos que são geralmente conflitantes entre si. Desta forma, algumas versões do recozimento simulado multiobjetivo têm sido propostas na literatura. Nessa seção será mostrada uma das versões mais recentes e eficientes. Estudos mais detalhados sobre Simulted Annealing multiobjetivo podem ser encontrados em (Ehrgott e Gandibleux (2000)), (Smith et al. (2004)), (Suman (2004)) e (Bandyopadhyay et al. (2008)).

A grande dificuldade em se obter uma versão eficiente multiobjetivo do recozimento simulado está no fato de que esse algoritmo gera somente um único ponto por execução, enquanto que algoritmos multiobjetivo devem idealmente gerar uma população de pontos não-dominados, que estejam espalhados pelo conjunto e que sejam próximos à fronteira Pareto Ótima. Para maiores detalhes sobre otimização multiobjetivo, vide o capítulo 17.

A solução encontrada para o problema de se gerar um conjunto de pontos no recozimento simulado consiste em armazenar os valores intermediários obtidos durante a execução da ferramenta em um conjunto de pontos não-dominados. Czyzak e Jaszkiewicz (1998) apresentaram uma primeira versão deste algoritmo e posteriormente Bandyopadhyay et al. (2008) apresentaram uma versão mais eficiente, cujos resultados são comparados com algoritmos consolidados na literatura, como o NSGA-II (Deb et al. (2002)) e PAES (Knowles e Corne (2000a)). Uma versão do recozimento simulado multiobjetivo é apresentada no Algoritmo 3, extraído de (Bandyopadhyay et al. (2008)).

Os parâmetros utilizados no Algoritmo 3 além das principais variáveis são detalhados na tabela 8.3.

O algoritmo apresentado em Bandyopadhyay et al. (2008) é chamado AMOSA (*Archived Multi-objective Simulated Annealing*), pois procura, a cada iteração, identificar os pontos não-dominados e armazená-los. É feita uma clusterização nesse conjunto de pontos não-dominados a fim de forçar a

Tabela 8.2: Parâmetros/variáveis e respectivos significados do recozimento simulado para problemas com variáveis contínuas.

Parâmetro / Variável	**Significado**
NT	Número de passos
ND	Número de ciclos auxiliares
LIM	Limite de movimentos com sucesso
j	j-ésimo ciclo auxiliar
i	i-ésima variável contínua
u_i	u-ésimo sucesso da i-ésima variável
TEMP	Temperatura
p_0	Melhor solução
C_i	$C_i \approx 0.2$
u	$u = \frac{\sum_i u_i}{\sum_i NT_i}$

Tabela 8.3: Parâmetros e respectivos significados do recozimento simulado para problemas multiobjetivo

Parâmetro / Variável	**Significado**
T_{max}	Temperatura máxima. Corresponde ao valor inicial da temperatura.
T_{min}	Temperatura mínima. Valor de referência para teste de convergência.
Archive	Conjunto de tamanho variável que irá armazenar os pontos durante a execução do SA. Esse conjunto deve ser inicializado com pontos no espaço de busca, gerados aleatoriamente. Ao final, esse conjunto deverá conter os pontos não-dominados obtidos.
HL	Tamanho máximo do conjunto *Archive* ao final da execução do algoritmo. Esse valor corresponde ao número de soluções não dominadas a serem encontradas.
SL	Tamanho máximo do conjunto *Archive* antes de ser aplicada a operação de clusterização de forma que o conjunto *Archive* tenha tamanho HL.
iter	Número máximo de iterações para cada temperatura.
α	Taxa de redução da temperatura.
Δdom_{min}	Variável referente ao menor valor de intensidade de dominância encontrado. Dado um subconjunto de k pontos, é calculado a intensidade de dominância entre *new_pt* e cada um desses k pontos, utilizando a equação 8.11. O menor desses valores é atribuído à variável Δdom_{min}.

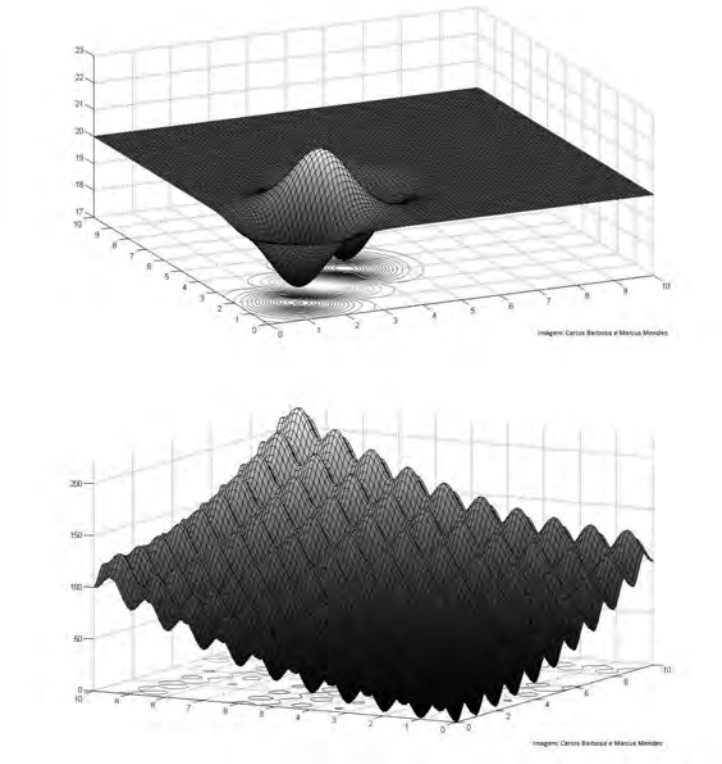


Figura 8.7: Função Exponencial e Rastrigin respectivamente.

diversidade das soluções. Durante a execução da ferramenta, o número de pontos não-dominados aumenta. Quando a quantidade de pontos atinge um valor ($|SL|$), é realizada a clusterização, utilizando-se $|HL|$ clusteres. O algoritmo de clusterização utilizado é Single Linkage (Jain e Dubes (1988)).

O algoritmo AMOSA utiliza um conceito a fim de mensurar um valor referente ao quanto uma solução é dominante sobre a outra. Esse conceito é chamado de *intensidade de dominância* e é definido da seguinte forma:

Definição 8.4 - *Sejam a e b duas soluções viáveis para um problema P. A intensidade de dominância é dada pela equação:*

$$\Delta dom_{a,b} = \prod_{i=1, f_i(a) \neq f_i(b)}^{M} \left(\frac{|f_i(a) - f_i(b)|}{R_i} \right), \tag{8.11}$$

onde M é o número de objetivos e R_i é o intervalo do i-ésimo objetivo.

O AMOSA procura por pontos que possuam uma grande intensidade de dominância. Entretanto, assim como o recozimento simulado mono-objetivo, nas primeiras iterações do algoritmo (enquanto a temperatura ainda está alta), são aceitos pontos com baixa intensidade de dominância e até mesmo pontos não-dominados. Contudo, à medida em que a temperatura vai caindo, a probabilidade de se aceitar um estado pior diminui.

Além de possuir uma complexidade computacional similar à complexidade dos algoritmos mais eficientes encontrados na literatura (vide tabela 8.4), resultados práticos mostraram que o AMOSA é

capaz de gerar resultados competitivos, considerando como medidas de comparação a:

- Convergência - Medida da distância entre os valores obtidos e a Fronteira Pareto Ótima obtida;
- Pureza - Fração calculada com a comparação entre o resultado obtido pelo algoritmo e a combinação dos resultados obtidos por diferentes algoritmos. Quanto mais próximo de 1, melhor é a performance; quanto mais próximo de 0, mais fraca é a performance do algoritmo em questão;
- Espaçamento - Medida da distribuição dos pontos não-dominados obtidos dentro da fronteira. Quanto maior for o espalhamento entre os pontos, melhor o resultado.

Tabela 8.4: Complexidade dos Algoritmos Multiobjetivos

Algoritmo	**Complexidade**
NSGA-II	$O(iter \times M \times N^2)$
PAES	$O(iter \times M \times N)$
AMOSA	$O(iter \times N \times (M + \log(N)))$

Na tabela 8.4 *iter* equivale ao número máximo de iterações (parâmetro que deve ser fornecido para o algoritmo); M é o número de objetivos e N é o tamanho da população. No caso do AMOSA, $N = |HL|$.

4. Conclusões

Esse capítulo apresentou o recozimento simulado, desde sua primeira versão mono-objetivo até a versão mais recente, multiobjetivo, destacando a sua aplicação em problemas discretos e contínuos. A simplicidade dessa ferramenta fez com que se tornasse bastante popular desde seu surgimento na comunidade científica, sendo utilizada em diversas aplicações.

Durante alguns anos, quando a comunidade de otimização se voltava mais para o estudo sobre problemas com múltiplos objetivos, o recozimento simulado foi pouco utilizado, uma vez que as primeiras versões multiobjetivo desse algoritmo não produziam um resultado com boa qualidade. A dificuldade inicial do recozimento simulado estava no fato de que só se trabalhava com um ponto por iteração, enquanto nas demais ferramentas Evolucionárias se trabalhava com uma população de pontos. A inserção da idéia de se usar um arquivo para armazenar os pontos não-dominados intermediários, bem como critérios para aumentar a diversidade (Pureza e Espaçamento) fez com que o recozimento simulado voltasse à pauta dos temas pesquisados como uma ferramenta robusta e eficiente para se resolver problemas de otimização.

Algoritmo 2 Pseudocódigo para o algoritmo recozimento simulado contínuo

1: INICIALIZA($NUCICL$, LIM, $TEMP$, x_0);
2: $Ind \leftarrow 0$; $H = \{h_1, h_2, \ldots, h_n\}$; $f_0 \leftarrow f(x_0)$;
3: $x'_i \leftarrow x'_i + rh_i$; $f' \leftarrow f(x')$; $NT_i \leftarrow NT_i + 1$;
4: **se** $f' \leq f$ **então**
5: $x \leftarrow x'$; $f \leftarrow f'$; $u_i \leftarrow u_i + 1$; $i \leftarrow i + 1$;
6: **se** $i > nd$ **então**
7: $i \leftarrow 1$;
8: **fim se**
9: **se** $f' < f_0$ **então**
10: $p_0 \leftarrow x'$; $f_0 \leftarrow f'$;
11: **fim se**
12: **senão**
13: {Execução do algoritmo de Metropolis}
14: Gere uma perturbação;
15: **se** Novo estado for aceito **então**
16: goto linha 3;
17: **fim se**
18: **fim se**
19: **se** $j < 10nd$ **então**
20: **se** $j > 2nd \wedge u \in [0.4; 0.6]$ **então**
21: goto linha 29;
22: **senão**
23: $j \leftarrow j + 1$;
24: **se** $Ind = 0 \wedge j \in [k, 2k, 3k]$ **então**
25: **se** $\frac{\sum_i u_i}{\sum_i NT_i} < 0.3$ **então**
26: Aumentar TEMP;
27: **fim se**
28: **se** $\frac{\sum_i u_i}{\sum_i NT_i} > 0.3$ **então**
29: Reduzir TEMP;
30: **fim se**
31: **senão**
32: goto linha 3;
33: **fim se**
34: **fim se**
35: **fim se**
36: **se** $\frac{u_i}{Nt_i} > 0.6$ **então**
37: $h_i \leftarrow h_i[1 + \frac{C_i(p_i - 0.6)}{0.4}]$;
38: **fim se**
39: **se** $\frac{u_i}{Nt_i} < 0.4$ **então**
40: $h_i \leftarrow h_i[1 + \frac{C_i(0.4 - p_i)}{0.4}]^{-1}$;
41: **fim se**
42: Reduzir TEMP; $Ind \leftarrow Ind + 1$;
43: **se** Atingiu Critério de Parada **então**
44: Fim do Algoritmo;
45: **senão**
46: $x \leftarrow p_0$; $f \leftarrow f_0$;
47: goto linha 3;
48: **fim se**

Algoritmo 3 Recozimento Simulado Multiobjetivo

INICIALIZA (T_{max}, T_{min}, HL, SL, $iter$, α, $temp$, $Archive$);
$current\_pt \leftarrow random(Archive)$;
enquanto $temp > T_{min}$ **faça**
  para $i \leftarrow 0$; $i < iter$; $i++$ **faça**
    $new\_pt \leftarrow$ perturb($current\_pt$);
    se $current\_pt$ domina new_pt **então** {Caso 1}
      $\Delta dom_{avg} = \frac{(\sum_{i=1}^{k} \Delta dom_{i,new\_pt}) + \Delta dom_{current\_pt,new\_pt}}{k+1}$; {$k$ = numero total de pontos em $Archive$ que dominam new_pt, $k \geq 0$}
      $prob = \frac{1}{1+\exp(\Delta dom_{avg} * temp)}$;
      Aceite new_pt como $current\_pt$ de acordo com a probabilidade $prob$;
    fim se
    se $current\_pt$ e new_pt não dominam um ao outro **então** {Caso 2}
      Verifique a dominância de new_pt em relação aos pontos em $Archive$
      se new_pt é dominado por $k, (k \geq 1)$ pontos de $Archive$ **então** {Caso 2a}
        $prob = \frac{1}{1+\exp(\Delta dom_{avg} * temp)}$; $\Delta dom_{avg} = \frac{\sum_{i=1}^{k} \Delta dom_{i,new\_pt}}{k}$;
        Aceite new_pt como $current\_pt$ de acordo com a probabilidade $prob$;
      fim se
      se new_pt é dominado por todos os pontos em $Archive$ **então** {Caso 2b}
        $new\_pt \leftarrow current\_pt$; Adicione new_pt a $Archive$;
        Se $Archive\_size > SL$, clusterize $Archive$ com HL clusteres;
      fim se
      se new_pt domina $k, (k \geq 1)$ pontos de $Archive$ **então** {Caso 2c}
        $new\_pt \leftarrow current\_pt$; Adicione new_pt a $Archive$; Remova os k pontos dominados de $Archive$;
      fim se
    fim se
    se new_pt domina $current\_pt$ **então** {Caso 3}
      Verifique a dominância de new_pt em relação aos pontos em $Archive$
      se new_pt é dominado por $k, (k \geq 1)$ pontos de $Archive$ **então** {Caso 3a}
        Δdom_{min} = MIN(diferença de dominância entre new_pt e os k pontos);
        $prob = \frac{1}{1+\exp(-\Delta dom_{min})}$;
        $current\_pt \leftarrow$ ponto que corresponde a Δdom_{min} com probabilidade = $prob$. Caso contrário, $new\_pt \leftarrow current\_pt$;
      fim se
      se new_pt é dominado por todos os pontos em $Archive$ **então** {Caso 3b}
        $new\_pt \leftarrow current\_pt$; Adicione new_pt a $Archive$;
        caso $current\_pt$ está em $Archive$, remova-o ponto, Senão, caso $Archive\_size > SL$, clusterize $Archive$ com HL clusteres;
      fim se
      se new_pt domina k pontos de $Archive$ **então** {Caso 3c}
        $new\_pt \leftarrow current\_pt$; Adicione new_pt a $Archive$; Remova os k pontos de $Archive$;
      fim se
    fim se
  fim para
  $temp = \alpha * temp$;
fim enquanto
Se $Archive\_size > SL$, clusterize $Archive$ com HL clusteres;