Otimização de circuitos usando Simulated Annealing

Vinicius Veroneze R. Costa[1], **Ivan S. Sendin**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Goiás
Campus Catalão (UFG)
75.704-020 – Catalão – GO – Brazil

veroneze@gmail.com, ivansendin@yahoo.com

Abstract. *In this paper, the Simulated Annealing optimization method is studied in a introductory way, seeking to build an initial idea about its use in modeling for circuit design optimization. Graphics and the resulting configuration are shown proving the efficiency, flexibility and reliability of the method.*

Resumo. *O método Simulated Annealing é estudado de forma introdutória neste artigo, buscando construir uma ideia inicial sobre seu uso e desenvolvimento numa modelagem para otimização de design de circuitos. São mostrados gráficos e a configuração resultado comprovando a eficácia, flexibilidade e confiabilidade do método.*

1. Introdução

Na metalurgia, annealing (recozimento) é o processo de aquecer um metal e em seguida resfriá-lo lentamente para reduzir seus defeitos, otimizando sua estrutura. Aquecer o metal faz com que os átomos movimentem-se aleatoriamente, mas controlando o aquecimento e resfriamento, aumenta-se a probabilidade de que os átomos se reorganizem em cristais maiores, com menor energia interna e, consequentemente, com menos defeitos.

A versão para computadores do Simulated Annealing imita a versão metalúrgica, determinando menores níveis de energia interna para a função objetivo, muitas vezes chamada de função custo. O algoritmo inicia com uma temperatura elevada que lentamente é reduzida enquanto se realizam modificações elementares, buscando uma aproximação do mínimo global da função objetivo.

Existem vários problemas de otimização que se tornam intratáveis quando se usam métodos combinatórios, principalmente se a quantidade de elementos apresenta grande dimensão. Para estes problemas, uma metaheurística para otimização estocástica, como Simulated Annealing, é altamente efetiva.

2. O Método

A execução do algoritmo Simulated Annealing ocorre da seguinte maneira:

- Dada uma configuração incial S_i
- Obtem-se a temperatura inicial T
- Enquanto o sistema não está solidificado, repita:
  - Modificação elementar em S_i obtendo S_j
  - Determinação da variação de energia ΔE

  - Regra de aceitação de Metropolis:
    Se $\Delta E \leq 0$, a modificação é aceita.
    Se $\Delta E \geq 0$, a modificação pode ser aceita com probabilidade $e^{\frac{-\Delta E}{T}}$
  - Redução paulatina de T
- Retorne a configuração otimizada.

De acordo com [Dreo 2006], a temperatura pode ser um simples elemento controlável do algoritmo ou pode-se criar uma função que a determine, baseada nos parâmetros do problema em questão. Na maioria dos casos, pode-se determinar a temperatura empiricamente sem perdas na qualidade de execução do algoritmo. Para a redução da temperatura, é recomendável que se aplique

$$T = \alpha * T, \mid 0,5 \leq \alpha \leq 0,9 \tag{1}$$

Para a modificação elementar em S_i, a variação de posição de um elemento aleatório pode ser suficiente, mas inverter a posição de dois elementos também pode ser uma boa escolha.

Para a determinação da variação de energia ΔE, primeiro define-se a função objetivo $f\,(s)$, que terá seu valor determinado com base em uma configuração passada como parâmetro. Assim, poderemos obter a variação de energia:

$$\Delta E = f\,(S_j) - f\,(S_i) \tag{2}$$

Quando $\Delta E \leq 0$, a modificação é aceita e faz-se $S_i = S_j$, mas quando $\Delta E \geq 0$, S_j só será aceito se $e^{\frac{-\Delta E}{T}} \geq R\,(0,1)$, onde $R\,(0,1)$ é um número real entre 0 e 1 gerado aleatoriamente. Note que quanto maior o valor de T, maior a probabilidade de S_j ser aceito.

3. O problema

Suponha um circuito com nxm elementos iguais e que cada elemento tenha v conexões com outros elementos. A função objetivo pode ser determinada como:

$$f = \sum_{i=0}^{q} \sum_{j=0}^{v} \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{3}$$

onde q é a quantidade de elementos do circuito e v é a quantidade de "vizinhos" de cada circuito, ou seja, a quantidade de elementos conectados ao elemento q, (x_i, y_i) são as coordenadas do elemento q e (x_j, y_j) são as coordenadas do vizinho v atual. Note que a função custo é simplesmente a soma da distância euclidiana entre os elementos e seus elementos vizinhos, ou seja, a soma do comprimento das conexões entre os componentes do circuito.

Uma aplicação para o método Simulated Annealing é a otimização de circuitos eletrônicos, para obter uma melhor disposição dos elementos e diminuir o custo com o material utilizado nas conexões entre eles.

Foi criado um circuito de 5 x 5 elementos, e com 80 conexões, conforme a (Figura 1). Os quadrados vermelhos representam os componentes do circuito e as linhas azuis representam as conexões.

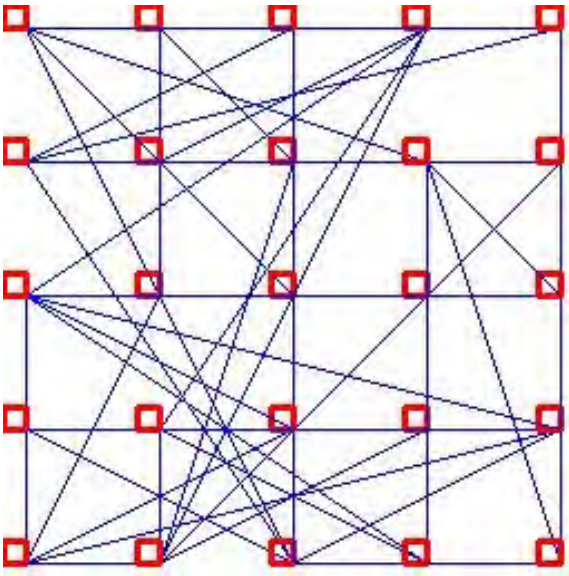

Figure 1. Circuito com a configuração inicial

Esta configuração inicial apresentou custo de 143,73 para as 80 conexões existentes, o que totaliza 1,797 de custo médio por conexão. Foi executado o algoritmo do Simulated Annealing com $T = 100,85$, $\alpha = 0,9$ e foi determinado que a configuração inicial estaria solidificada após 10.000 iterações. Em aproximadamente 18 segundos o algoritmo retornou uma configuração otimizada.

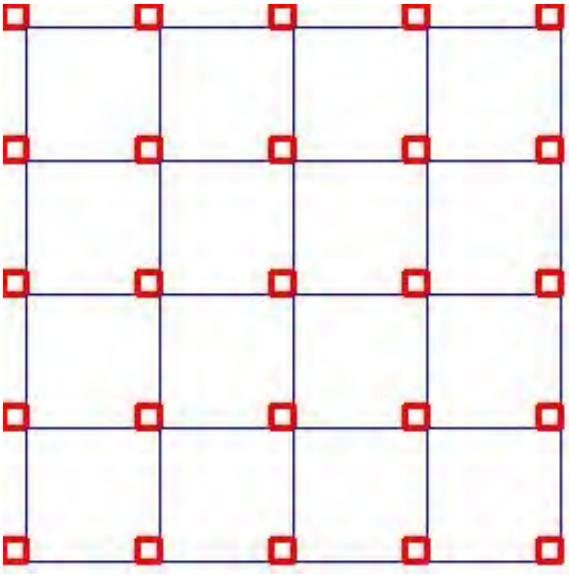

Figure 2. Circuito com a configuração otimizada

Na configuração otimizada, o custo total foi de 80, levando a 1 como custo médio. Perceba que esta configuração se trata de um mínimo global, e não somente um mínimo local ou uma aproximação ao mínimo global (Figura 2).

Um gráfico da execução é mostrado na (Figura 3).

É interessante observar que nas primeiras iterações ocorre um aumento na função

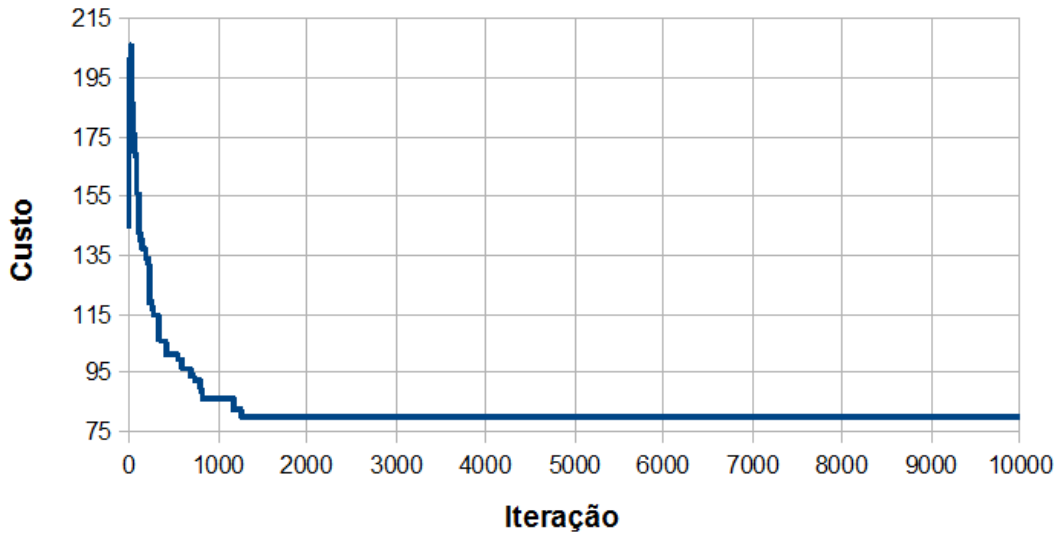

Figure 3. Gráfico Iteração x Custo

custo devido ao fato de os valores de T ainda serem altos. Esses valores aumentam a probabilidade de ser aceita uma solução que não possua custo menor que o atual, ou seja, $e^{\frac{-\Delta E}{T}} \geq R(0,1)$ será verdadeiro mais vezes para valores maiores de T. Detalhou-se o início da execução do algorimo na (Figura 4).

Até a 10ª iteração o algoritmo seleciona uma configuração com custo mais alto que o inicial. Entre as iterações 10 e 40 são escolhidas configurações com custos menores, mas que são descartadas em detrimento a configurações com custo maior. Isso se trata da capacidade do algoritmo de fugir dos mínimos locais, na busca pela aproximação ao mínimo global.

Em circuitos de maior dimensão, o algoritmo retornou mais vezes uma configuração com custo aproximado ao mínimo global da função ao invés do mínimo global exato, como na configuração com 5x5 elementos. Em uma das execuções, foi gerado um circuito 10x10 (100 elementos, 360 conexões) que apresentou custo 1734. Após a aplicação do Simulated Annealing, foi retornada uma configuração com custo de 421, que representa 1,17 de custo por conexão. O mínimo global retonaria um custo por conexão com valor 1,0. A execução do algoritmo, nesse caso, se deu em 73 segundos.

4. A implementação

Apesar de não haver grandes complicações para a implementação do algoritmo, alguns desafios surgiram e relatam-se a seguir.

Foi escolhida a linguagem Python para a implementação, principalmente pela sua capacidade de manipulação de listas. Na (Figura 5) está o código da função que realiza o Simulated Annealing.

A função recebe 4 parâmetros:

- a é a configuração inicial do circuito

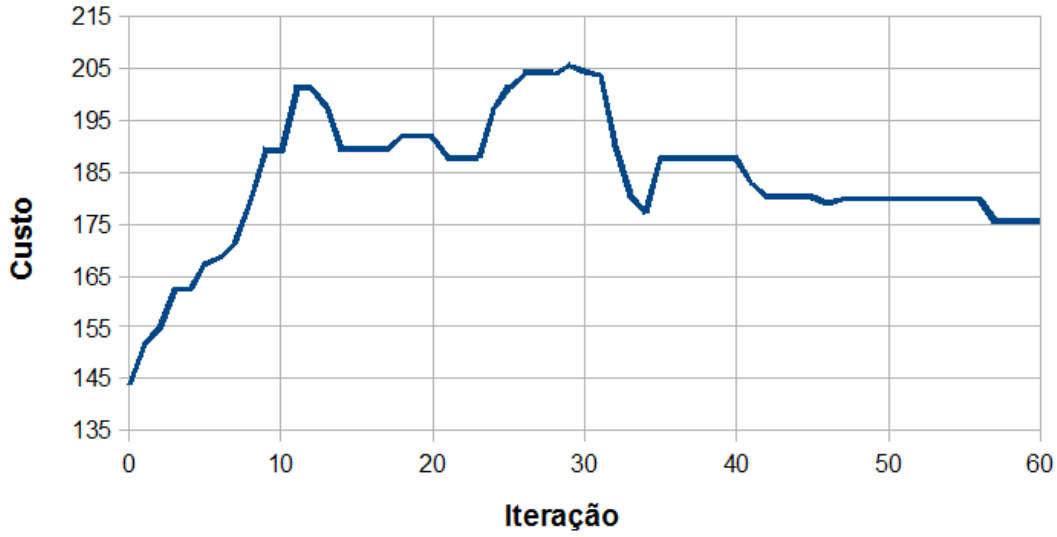

Figure 4. Primeiras 60 iterações

- t é a temperatura inicial
- n é a quantidade de iterações
- alfa é o fator de decremento da temperatura

O algoritmo repete por n iterações, calculando o delta, que é a diferença entre o custo da configuração modificada e o custo da melhor configuração atual. Caso delta seja menor que 0, significa que a configuração modificada é melhor que a atual, então temos uma nova melhor configuração. Caso delta não seja menor, se $e^{\frac{-\Delta E}{T}} \geq R(0,1)$, a configuração é aceita mesmo tendo custo maior. Em seguida, t é decrementado proporcionalmente a alpha. Ao final, o algoritmo retorna a melhor configuração encontrada.

Note na linha 8 a chamada à função perturba. Essa função é a responsável por trocar a posição de 2 elementos aleatórios da configuração.

É importante ressaltar o uso constante de copy.deepcopy(). Como o Python trabalha com referências, principalmente com as listas, que são objetos, fazer Si = Sj está criando uma referência somente, ou seja, quando chama-se perturba(Sj), na verdade Si está sendo passado como argumento, o que não deixará outra configuração para comparar o custo, por isso a necessidade do uso de copy.deepcopy(). Apesar do alto custo computacional dessa função, o algoritmo foi executado rapidamente.

5. Conclusão

Neste trabalho buscou-se introduzir o algoritmo do Simulated Annealing, usando uma aplicação em otimização de design de circuitos. Foi observado que ele consegue encontrar o mínimo global da função custo na maioria das vezes, bastando ajustar os argumentos de entrada, principalmente a temperatura e a quantidade de iterações.

Nos próximos trabalhos pretende-se desenvolver o Simulated Annealing para refinamento de proteínas, com estrutura tridimensional e aplicações na Bioinformática. Um

```
def sa(a, t, n, alfa):

    Si = copy.deepcopy(a)
    Sj = copy.deepcopy(a)

    for i in range(n):
        ci = getCusto(Si)[0]
        Sj = copy.deepcopy(perturba(copy.deepcopy(Si)))
        cj = getCusto(Sj)[0]

        delta = cj - ci

        if delta < 0:
            Si = copy.deepcopy(Sj)

        else :
            r = random.random()

            if math.exp(-delta / t) > r:
                Si = copy.deepcopy(Sj)

        t = t * alfa

    return Si
```

Figure 5. Código da função principal do Simulated Annealing

dos primeiros passos será o desenvolvimento de uma função para determinar a temperatura inicial do modelo para substituir a temperatura empírica que utilizou-se nesse trabalho.

References

Dreo, J. (2006). *Metaheuristcs for Hard Optimization*. Springer.

Gielen, G., Walscharts, H., and Sansen, W. (1990). Analog circuit design optimization based on symbolic simulation and simulated annealing. *Solid-State Circuits, IEEE Journal of*, 25(3):707 –713.

Ji-Yang, Q. (2010). Application of improved simulated annealing algorithm in facility layout design. In *Control Conference (CCC), 2010 29th Chinese*, pages 5224 –5227.

Kumar, M. (2012). An information guided framework for simulated annealing. In *American Control Conference (ACC), 2012*, pages 827 –832.