ESTUDO COMPARATIVO DE HEURÍSTICAS PARA A RESOLUÇÃO DE PROBLEMAS COMBINATÓRIOS

José Francisco Ferreira Ribeiro
Universidade de São Paulo
USP – ICMC-SCE Av. Dr. Carlos Botelho, 1465 – 13560-970 São Carlos SP
jffr@icmc.usp.br

Ivan Jeukens
Universidade de São Paulo
USP – EP – LME Av. Prof. Luciano Gualberto, 3, 158 – 05508-900 São Paulo SP
ijeukens@lme.usp.br

Resumo

Um estudo comparativo entre a "busca tabu" e o "simulated annealing" para a resolução de problemas combinatórios é apresentado nesse artigo. Os problemas tratados foram os seguintes: Caixeiro-Viajante, Atribuição Quadrática e Alocação em Circuitos. Os conceitos fundamentais das duas técnicas são apresentados e discutidos. Os testes realizados demonstraram que a "busca tabu" e o "simulated annealing" podem resolver esses problemas em tempo computacional baixo, obtendo a solução ótima ou uma solução de boa qualidade em poucos segundos.

Palavras-Chaves: Otimização; Metaheurísticas; Combinatória; Busca Tabu; Simulated Annealing.

Abstract

A comparative study between simulated annealing and tabu search for solving combinatorial problems is presented in this paper. The problems treated are the Traveling Salesperson, the Quadratic Assignment Problem and the Placement Problem. The fundamental concepts of simulated annealing and tabu search are presented and discussed. Tests carried out demonstrated that simulated annealing and tabu search are able to solve these problems in very short computational time, obtaining a solution, which is either optimal, a high standard or feasible in some seconds.

Keywords: Optimization; Meatheuristics; Combinatorics; Tabu Search; Simulated Annealing.

1. INTRODUÇÃO

Um problema de otimização combinatória é descrito por um par (Ω, f_c), onde Ω representa o conjunto de todas as soluções para o problema e f_c é a função custo. Para um dado problema, procura-se a solução de menor custo entre todas as soluções do conjunto Ω. Esta solução é denominada de ótimo global e não é necessariamente única. Devido à complexidade computacional [5] associada a muitos dos problemas de otimização combinatória, os algoritmos exatos, i.e., que obtêm o ótimo global para uma instância do problema, são apenas praticáveis para instâncias de tamanho limitado, longe dos problemas reais. Assim, lança-se mão de algoritmos aproximados, ou heurísticas, para a obtenção de uma solução próxima do ótimo em um tempo de cálculo limitado. Entre estas heurísticas, destacam-se os algoritmos simulated annealing e tabu search, baseados na ativação de uma busca em torno da vizinhança de uma solução.

2. SIMULATED ANNEALING (SA)

O algoritmo simulated annealing (SA) foi desenvolvido independentemente por [4] e [7]. Ele baseia-se na analogia entre a simulação do resfriamento de sólidos e a resolução de problemas de otimização combinatória. O termo annealing denota o processo físico onde um sólido é submetido a uma elevação de temperatura até tornar-se líquido e com moléculas dispostas aleatoriamente. A partir deste estado, a temperatura é então lentamente diminuída e todas as partículas vão se organizando em grupos de mínima energia. A diminuição da temperatura deve ser feita de maneira apropriada: a cada temperatura, o sólido deve poder alcançar o equilíbrio térmico, que é caracterizado pela probabilidade de ele estar em um estado de energia e dado pela distribuição de Boltzmann: $P(E = e) = 1 / Z(T)^{exp(-e/kbT)}$ onde Z(T) é um fator de normalização e kb é a constante de Boltzmann. Na medida em que a temperatura é diminuída, a distribuição de Boltzmann concentra-se em estados de menor energia, até que com a temperatura próxima de zero, apenas os estados de energia mínima possuem uma probabilidade diferente de zero ocorrer.

2.1. CRITÉRIO DE METROPOLIS

Com o objetivo de simular o comportamento de um sólido evoluindo para o equilíbrio térmico a uma dada temperatura, Metropolis et al. [8] investigaram o método de Monte Carlo. Este método gera uma seqüência de estados que eventualmente irão atingir o equilíbrio térmico. Um novo estado é determinado a partir do estado atual, que é caracterizado pelo posicionamento das partículas do sólido, provocando-se de forma aleatória uma pequena perturbação em uma partícula selecionada aleatoriamente. A diferença entre a energia do novo estado e do estado atual é determinada. Caso essa seja negativa, o novo estado é aceito, ou seja, ele passa a ser o estado atual. Caso contrário, a probabilidade do novo estado ser aceito é dada por $\exp(-\Delta E/Kb\ T)$ e esta regra de aceitação é conhecida por critério de Metropolis e pode ser utilizado na resolução de problemas de otimização combinatória. Neste caso, os vários estados do sólido representam as soluções factíveis do problema, a temperatura é denominada de parâmetro de controle, a constante de Boltzmann não é utilizada e a energia do sistema é representada pela função-objetivo.

2.2. ALGORITMO

O algoritmo simulated annealing pode então ser descrito como uma seqüência de execuções do algoritmo de Metropolis com o parâmetro de controle diminuído a cada execução [11]. Genericamente, o algoritmo pode ser expresso pelo seguinte pseudocódigo:

```
algoritmo simulated annealing
        inicializações (solução inicial aleatória e custo);
        iter := 0; t := temperatura_inicial;
        repetir
                repetir
                        perturbar (configuração_atual, ΔC);
                        se o critério de Metropolis for satisfeito então
                                aceitar a nova configuração;
                até estar próximo do equilíbrio
                t_iter+1 := f(iter);  iter := iter + 1;
        até critério_de_parada = tue
fim algoritmo simulated annealing
```

O algoritmo é constituído de 2 nested loops. O loop interno corresponde ao algoritmo de Metropolis, onde a solução atual é perturbada, a variação do valor da função de custo é

calculada e a solução é aceita ou rejeitada. O critério de Metropolis é dado por $\Delta C < 0$ ou rand(0, 1) < exp(-ΔC/t), onde ΔC é o custo da solução perturbada subtraído do custo da solução atual e rand(0, 1) é responsável pela geração de um número real entre 0 e 1. A constante de Boltzmann não é utilizada, pois a temperatura é apenas um parâmetro de controle sem conotação física. Uma vez que a solução esteja em equilíbrio térmico, o loop interno é encerrado. Então, a temperatura é atualizada e todo o processo é repetido até que um dado critério de parada seja igual a true. O critério de Metropolis aceita soluções que melhoram e pioram o custo da solução atual, pretendendo assim escapar dos ótimos locais.

2.3. COOLING SCHEDULE

A parte fundamental para o sucesso do algoritmo simulated annealing é a determinação dos parâmetros conhecidos como cooling schedule:

1) valor da temperatura inicial;
2) função de diminuição da temperatura;
3) número de perturbações em cada temperatura;
4) critério de parada. Existem 2 tipos de cooling schedule: estático e dinâmico.

O tipo estático determina os parâmetros e não os modifica durante a execução do algoritmo. O tipo dinâmico especifica os parâmetros com base em dados coletados durante a execução do algoritmo. A descrição destes parâmetros para o TSP e o QAP é feita abaixo, enquanto que o PLA utiliza aqueles propostos em [8].

2.3.1. Temperatura Inicial

Uma regra geral para determinação da temperatura inicial consiste em utilizar uma temperatura em que praticamente todas as soluções de uma cadeia sejam aceitas. Alguns estudos determinaram que esse valor é superior ou igual ao desvio padrão dos custos em todo o espaço de soluções (σ_∞).

Como é impraticável determinar este valor de forma exata, uma boa aproximação é obtida analisando-se 1000 soluções aleatoriamente [13]. Para os 2 casos de schedule, a temperatura inicial será igual a σ_∞.

2.3.2. Função de Diminuição da Temperatura

O schedule estático [7] utiliza uma função de diminuição da temperatura cujos parâmetros são determinados de forma fixa: $t_k = t_{ini} \cdot \alpha^k$.

O parâmetro α é geralmente 0.95 pois um valor menor faz com que a temperatura final seja rapidamente alcançada.

O schedule dinâmico [1] utiliza a seguinte função: $t_k = t_{k-1} \ [1 + t_{k-1}.\ln(1+\delta) / 3\ \sigma_{k-1}]^{-1}$, onde:

– σ_{k-1} é o desvio padrão das soluções geradas na temperatura t_{k-1}

– δ é uma constante com valor positivo e pequeno, geralmente .085.

2.3.3. Número de Perturbações

O número de perturbações, i.e., soluções geradas em dada temperatura, afeta diretamente a qualidade da solução obtida e o tempo de execução. Ele também está diretamente relacionado com a função de redução da temperatura, uma vez que quanto mais

rápida a diminuição de temperatura, maior será o número de soluções geradas para que o equilíbrio seja alcançado.

Um valor tido como apropriado para perturbações nos 2 casos de schedule é $|N(x, \sigma)|$, ou seja, a cardinalidade do conjunto de soluções vizinhas de uma dada solução.

2.3.4. Critério de Parada

O critério de parada é baseado no fato de que quando a melhora na qualidade da solução é sucessivamente pequena o algoritmo deve interromper-se. O schedule estático interrompe a execução quando 3 soluções sucessivas obtiverem o mesmo valor. O schedule dinâmico utiliza a seguinte expressão: $[\sigma^2 / t_f (\mu_0 - \mu_f)] < \theta$, onde θ é uma constante pequena e positiva, em geral .00001 e μ_k é a média dos custos das soluções geradas na temperatura k.

2.3.5. Two-Stage Simulated Annealing (TSSA)

Para contornar o alto tempo de execução exigido pelo algoritmo simulated annealing, [6] e [12] propuseram a seguinte técnica:

Inicialmente, executa-se uma heurística de otimização local e, em seguida, uma temperatura é calculada com base no custo desta solução.

2.3.6. Critério de Varanelli

Varanelli [13] observou que a forma de estudar o comportamento da melhor solução (bsf) consiste em normalizar o valor de bsf em unidades de desvio padrão (σ_∞) com relação ao custo estimado (E_∞) de todas as soluções.

Tem-se, então, as equações: $bsf_{norm} = [(E_\infty - bsf) / \sigma_\infty]$ e $t_{norm} = (t_k / t_0)$, onde t_0 é a temperatura inicial.

Estudos empíricos mostraram a necessidade de se calcular o desvio existente entre o custo esperado e o custo bsf, e [13] propôs que o cálculo do desvio γ_∞ deveria ser feito de forma probabilística, utilizando-se a seguinte equação:

$$P[E_\infty - \gamma_\infty \sigma_\infty < X < E_\infty + \gamma_\infty \sigma_\infty] \approx |\text{número\_de\_perturbações}|^{-1}.$$

3. TABU SEARCH (TS)

O algoritmo tabu search [9] pode ser considerado uma extensão genérica dos algoritmos de otimização local. Ele faz uso da memorização de soluções geradas durante a execução como forma de escapar dos ótimos locais e evitar que soluções previamente obtidas sejam repetidas.

A heurística não utiliza métodos aleatórios como é o caso da heurística annealing. Entretanto, da mesma forma que o algoritmo simulated annealing, a tabu search está baseada no conceito de vizinhança.

3.1. ALGORITMO

O pseudocódigo abaixo ilustra o procedimento:

```
algoritmo tabu search
        inicializações;
        enquanto condição_de_parada = false
                analisar um conjunto de soluções;
                escolher a melhor solução;
```

```
                solução_atual := solução_escolhida;
                atualizar_as_memórias (solução_escolhida);
        fim enquanto
fim algoritmo tabu search
```

O conjunto de soluções analisadas é composto pelas soluções da estrutura de vizinhança da solução atual. A escolha da melhor solução é feita da seguinte forma: 1) observar a solução com maior redução da função custo; 2) caso a solução não seja tabu, então esta solução é utilizada; 3) caso a solução seja tabu, verifica-se se ela satisfaz o critério de aspiração; 4) caso a solução satisfaça o critério de aspiração, esta solução é utilizada; 5) caso a solução não satisfaça o critério de aspiração, elimina-se a solução e retorna-se o processo de escolha a partir do item 1. O critério de escolha da melhor solução tem 2 momentos cruciais: 1) decidir quando uma solução é tabu; 2) qual o critério de aspiração.

A decisão de quando uma solução é tabu é feita com base em uma memória, armazenando-se, por exemplo, a transformação que gerou a nova solução, bem como quando ela ocorreu. Caso a mesma transformação ocorra dentro de um certo intervalo de tempo, a nova solução é considerada tabu. O intervalo de tempo é chamado de tamanho da lista tabu, uma vez que a memória para armazenar as transformações pode ser implementada como uma lista. O critério de aspiração constitui-se em comparar o custo da nova solução com o custo da melhor solução encontrada até o momento. Caso o custo da nova solução seja inferior, esta não é considerada tabu. Deve observar-se que a tabu search permite que soluções que aumentem o custo sejam aceitas. Desta forma, procura-se escapar dos ótimos locais.

3.2. REACTIVE TABU SEARCH

A heurística Reactive Tabu Search [2] utiliza uma memória extra para armazenar a solução aceita ou seu custo. Com base nesta informação, o método procura de forma explícita evitar a repetição de soluções e o confinamento do algoritmo a apenas uma solução do conjunto de soluções. O pseudocódigo abaixo ilustra o procedimento:

```
algoritmo reactive tabu search
inicializações;
enquanto condição_de_parada = false
        analisar um conjunto de soluções;
        escape := verificar_repetições (solução_atual);
        se escape = true        então randomizações;
        senão           escolher a melhor solução;
                        solução_atual := solução_escolhida;
                        atualizar_as_memórias (solução_escolhida);
fim enquanto
fim algoritmo reactive tabu search
```

Este pseudocódigo é similar ao anterior, salvo a utilização da subrotina verificar_repetições. Esta subrotina é encarregada de verificar se a solução atual já ocorreu anteriormente durante a execução. Se ela não tiver ocorrido, o algoritmo prossegue normalmente. Senão, a subrotina verifica um contador interno que armazena quantas vezes tal solução ocorreu. Se o valor do contador estiver abaixo de uma certa constante, então apenas o tamanho da lista tabu é aumentado. Do contrário, a subrotina retorna um valor booleano true, que ativa alterações aleatórias na solução.

4. PROBLEMAS TESTADOS

Foram escolhidos 3 problemas para um estudo comparativo entre o SA e a TS:

1) TSP – Traveling Salesperson;
2) QAP – Quadratic Assignment Problem;
3) PLA – Placement.

O TSP deve encontrar o caminho mínimo que passe uma e uma única vez em todos os nós de um grafo. Utilizou-se a versão simétrica do problema, i.e., a distância do nó i ao nó j é igual à distância do nó j ao nó i. O QAP minimiza o custo da função: $f(\phi) = \Sigma \Sigma a_{ij} b_{\phi(i)\phi(j)}$. A é a matriz de distâncias entre 2 localidades i e j, e B representa o fluxo entre 2 localidades i e j. A versão utilizada da matriz A é simétrica. O PLA tratado neste estudo é o modelo para o problema de placement em circuitos do tipo reprogramável por memória RAM [3].

5. RESULTADOS EXPERIMENTAIS

Para a realização dos testes, foi utilizada a biblioteca TSPLIB95 [10]. Todas as medidas de tempo (segundos) foram feitas em um microcomputador Pentium 200 MHz com 64 Mbytes, sistema operacional Linux K2.0.0 e compilador C gcc, opção –O3.

5.1. TSP – TRAVELLING SALESPERSON

As Tabelas 1, 2, 3, 4 e 5 fornecem os resultados para as instâncias selecionadas, respectivamente, para as seguintes opções:

1) SA, schedule estático, sem TSSA;
2) SA, schedule dinâmico, sem TSSA;
3) SA, schedule estático, com TSSA;
4) TS;
5) RTS.

Instância	Custo	Tempo
Dantzig42	709.8	0.356
Brazil58	25411	0.973
KroA100	21568.2	3.147
Lin105	14556.2	3.477
D198	15939.6	16.68
Tsp225	4012.4	21.06
Rd400	15633.8	92.83

Tabela 1

Instância	Custo	Tempo
Dantzig42	702.8	1.436
Brazil58	25492.2	4.055
KroA100	21460.2	18.48
Lin105	14444.8	21.77
D198	15891.4	147.8
Tsp225	3947.4	188.5
Rd400	15552	1004

Tabela 2

Instância	Custo	Tempo
Dantzig42	701.6	0.2772
Brazil58	25502.8	0.745
KroA100	21610	2.193
Lin105	14546.2	2.4
D198	16021.6	11.08
Tsp225	3976.8	12.57
Rd400	15773.6	54.14

Tabela 3

Instância	Custo	Tempo
Dantzig42	872	0.029
Brazil58	35217	0.1
KroA100	32649	0.056
Lin105	29089	0.752
D198	39596	6.611
Tsp225	7698	7.456
Rd400	36604	57.77

Tabela 4

Instância	Custo	Tempo
Dantzig42	863.6	0.041
Brazil58	34270.4	0.2919
KroA100	37660	0.53
Lin105	31355	0.613
D198	41360	6.262
Tsp225	8524.4	7.1826
Rd400	38398.2	65.67

Tabela 5

5.2. QAP - QUADRATIC ASSIGNMENT PROBLEM

As Tabelas 6, 7, 8 e 9 fornecem os resultados para as instâncias selecionadas, respectivamente, para as seguintes opções:

6) SA, schedule estático, sem TSSA;

7) SA, schedule estático, com TSSA;

8) TS;

9) RTS.

Instância	Custo	Tempo
Chr18a	15086	0.046339
Chr25a	4914.8	0.145339
Had16	3721.6	0.012613
Had20	6350.6	0.065133
Tai40	3248142.8	0.415824
Tai50	5115602.8	0.468601
Tai75	12160787.2	2.768244

Tabela 6

Instância	Custo	Tempo
Chr18a	15243.5	0.1404428
Chr25a	5141.2	0.31519
Had16	3721	0.105598
Had20	6924.8	0.216762
Tai40	3239803.2	0.532601
Tai50	5094001.6	1.403622
Tai75	12184668.4	3.783508

Tabela 7

Instância	Custo	Tempo
Chr18a	15580	0.008
Chr25a	5090.8	0.021
Had16	3721.2	0.007
Had20	6989.2	0.012
Tai40	3235657	0.089
Tai50	5083700	0.179
Tai75	12189270	0.492

Tabela 8

Instância	Custo	Tempo
Chr18a	15522.8	0.008
Chr25a	5304.4	0.019
Had16	3731.6	0.011
Had20	6986.8	0.018
Tai40	3232622.8	0.09
Tai50	5120162.4	0.173
Tai75	12206107.2	0.584

Tabela 9

5.3. PLA – PLACEMENT PROBLEM

As Tabelas 10 e 11 fornecem os resultados para as instâncias selecionadas, respectivamente, para as seguintes opções:

10) SA, sem TSSA;
11) TS.

Instância	Custo	Tempo
I1	1.545	4.149
Cu	1.53	2.656
9syma	1.146	1.675
Apex7	8.756	21.52
Alu2a	12.15	16.25
C499a	8.399	15.41
C880a	12.07	25.81
Cordic	48.67	108.38

Tabela 10

Instância	Custo	Tempo
I1	1.8	0.1
Cu	1.75	0.06
9syma	1.21	0.02
Apex7	10.73	1.259
Alu2a	14.04	1.19
C499a	9.91	0.939
C880a	14.55	1.77
Cordic	49.12	52.52

Tabela 11

6. COMENTÁRIOS E CONCLUSÃO

Este artigo apresentou um estudo comparativo de desempenho entre as metaheurísticas:

- simulated annealing
- tabu search

para a resolução dos seguintes problemas dos seguintes problemas de otimização combinatória:

- Traveling Salesperson
- Quadratic Assignment Problem
- Placement Problem

Nos testes realizados, o simulated annealing mostrou-se mais eficiente que a tabu search, em termos da qualidade da solução obtida. Em relação ao tempo computacional, a tabu search foi expressivamente mais rápida A técnica TSSA foi bem sucedida na resolução do TSP, reduzindo o tempo de cálculo praticamente pela metade. Para todas as instâncias e opções algoritmicas testadas, tanto o simulated annealing quanto a tabu search mostraram-se rápidos e capazes de fornecer soluções factíveis, ε-ótimas ou ótimas em tempo de cálculo perfeitamente aceitável.

7. AGRADECIMENTOS

Esse trabalho foi realizado com o auxílio do CNPq.

8. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Aarts, E. H. L., Laarhoven, P. J. M., "A New Polynomial-Time Cooling Schedule", *Proc. IEEE ICCAD-85*, Santa Clara, USA, pp. 206-208, 1985.

[2] Battiti, R., Tecchiolli, G., "The Reactive Tabu Search", *ORSA Journal on Computing*, 6, 2, pp. 126-140, 1994.

[3] Benz, V., Rose, J., "VPR: A New Packing, Placement and Routing Tool for FPGA Research", *International Workshop on Field Programable Logic*, 1997.

[4] Cerny, V., "Thermodynamical Approach to the TSP: An Efficient Simulation Algorithm", *J. Optimization Th. And Appl.*, 45, pp. 41-51, 1985.

[5] Garey, M. R., Johnson, D. S., "Computers and Intractability: A Guide to the Theory of NP-Completeness"

[6] Grover, L. K., "Standard Cell Placement Using Simulated Sintering", *Proc. 24th ACM/IEEE DAC*, Miami, USA, pp. 56-59, 1987.

[7] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., "Optimization by Simulated Annealing", *Science*, 220, pp. 45-54, 1983.

[8] Metropolis, N., Rosenbluth, A. W., Teller, A. H., "Equation of State Calculation by Fast Computing Machines", *J. of Chem. Phys.* , 21, pp. 1087-1091, 1953.

[9] Reeves, C. R., "Modern Heuristic Techniques for Combinatorial Problems", John Wiley, 1993.

[10] Reinelt, G., "TSPLIB 95", Universität Heidelberg, Alemanha.

[11] Romeo, F., Vicentelli, A. S., "A Theoretical Framework for Simulated Annealing", Algorithmica, 6, pp. 302-345, 1991.

[12] Rose, J. S., Snelgrove, W. M., Vranesic, Z. G., "Parallel Standard Cell Placement with Quality Equivalent to Simulated Annealing", *IEEE Trans. CADICS*, 7, pp. 387-396, 1988.

[13] Varanelli, J. M., "On the Acceleration of simulated annealing", Ph.D. Thesis, University of Virginia, USA, 1996.