

Algoritmo Metropolis

Metropolis, Rosenbluth, Rosenbluth, Teller, Teller'53

- ▶ Simula um sistema físico, de acordo com princípios da mecânica estatística.
- ▶ Baseado em passos de buscas locais.
- ▶ Passos buscando melhorar a solução, mas pode obter soluções piores para sair de ótimos locais, com alguma probabilidade.
- ▶ Usa conceito de temperatura e estados de energia (Gibbs-Boltzmann).

Sem perda de generalidade, vamos supor problemas de minimização

Função de Gibbs-Boltzmann

- ▶ A probabilidade de encontrar um sistema físico em um estado de energia E é proporcional a

$$e^{-\frac{E}{kT}},$$

onde $T > 0$ é uma temperatura e k é uma constante.

- ▶ Para qualquer temperatura $T > 0$, a função é monotonicamente decrescente em E .
- ▶ Sistema tende a estar em um estado de baixa energia
 - ▶ T grande: Estados de energia alta e baixa tem basicamente mesma chance
 - ▶ T pequeno: favorece estados de baixa energia

Algoritmo Metropolis

Notação:

\mathcal{N} = Conjunto de estados (no grafo de vizinhanças)

$\mathcal{N}(S)$ = Conjunto de estados vizinhos a S (no grafo de vizinhanças)

$E(S)$ = nível de energia do estado S (custo de uma solução)

k = Parâmetro para calibrar estados (soluções)

T = Temperatura (controla busca por vizinhos piores/melhores)

- ▶ Considera temperatura T fixa
- ▶ Mantém um estado corrente S durante a simulação
- ▶ Para cada iteração, obtém novo estado $S' \in \mathcal{N}(S)$
- ▶ Se $E(S') \leq E(S)$, atualiza estado corrente para S'
- ▶ Caso contrário, atualiza estado corrente com probabilidade $e^{-\frac{\Delta E}{kT}}$, onde $\Delta E = E(S') - E(S) > 0$.

Algoritmo Metropolis

ALGORITMO METROPOLIS - MINIMIZAÇÃO

1. Seja k e T (temperatura) parâmetros para calibrar problema
2. Obtenha uma solução inicial S
3. $S^+ \leftarrow S$ (mantém a melhor solução obtida)
4. Enquanto não atingir condição de parada, faça
 5. Seja $S' \in \mathcal{N}(S)$ solução vizinha escolhida aleatoriamente
 6. $\Delta S \leftarrow c(S') - c(S)$
 7. Se $\Delta S \leq 0$ então
 8. $S \leftarrow S'$
 9. Se $c(S) < c(S^+)$ faça $S^+ \leftarrow S$
 10. senão
 11. com probabilidade $e^{-\frac{\Delta S}{kT}}$ faça $S \leftarrow S'$
12. Devolva S^+

Algoritmo Metropolis

Teorema:

Seja $f_S(t)$ a fração das t primeiras iterações onde a simulação percorre S . Então, assumindo algumas condições técnicas, com probabilidade 1, temos

$$\lim_{t \rightarrow \infty} f_S(t) = \frac{1}{Z} e^{-\frac{E(S)}{kT}},$$

onde

$$Z = \sum_{S \in \mathcal{N}} e^{-\frac{E(S)}{kT}}.$$

Intuição: A simulação gasta basicamente o tempo correto em cada estado, de acordo com a equação de Gibbs-Boltzmann.

Simulated Annealing

- ▶ T grande \Rightarrow probabilidade de aceitar uma solução pior é grande (movimentos de subida do morro)
- ▶ T pequeno \Rightarrow probabilidade de aceitar uma solução pior é pequena.

Analogia física: Não é esperado obter um sólido com estrutura cristalográfica boa se

- ▶ o colocarmos em alta temperatura
- ▶ estiver quente e congelarmos abruptamente

Annealing: Cozimento de material em alta temperatura, permitindo obter equilíbrio gradual sucessivo em temperaturas menores.

Idéia: Controlar o algoritmo Metropolis variando a temperatura

Simulated Annealing - Simplificado

SIMULATED ANNEALING - MINIMIZAÇÃO

1. Faça $T \leftarrow T_0$ (temperatura inicial)
2. Seja S solução inicial
3. $S^+ \leftarrow S$ (mantém a melhor solução obtida)
4. Enquanto não atingiu condição de parada faça (loop externo)
5. Enquanto não atingir condição de parada, faça (loop interno)
6. $S' \leftarrow$ Obtenha-Solução-Vizinha(S)
7. $\Delta S \leftarrow c(S') - c(S)$
8. Se $\Delta S \leq 0$ então
9. $S \leftarrow S'$
10. Se $c(S) < c(S^+)$ faça $S^+ \leftarrow S$
11. senão
12. com probabilidade $e^{-\frac{\Delta S}{kT}}$ faça $S \leftarrow S'$
13. Atualiza-Temperatura(T)
14. Devolva S^+

Simulated Annealing

Possíveis implementações das subrotinas:

OBTENHA-SOLUÇÃO-VIZINHA(S)

- ▶ Aleatória: Escolha vizinho $S' \in \mathcal{N}(S)$ aleatoriamente
- ▶ Vizinho-Melhor: Escolha $S' \in \mathcal{N}(S)$ com $c(S')$ mínimo

ATUALIZA-TEMPERATURA(T)

- ▶ Esfriamento geométrico:
- ▶ Faça $T \leftarrow \alpha \cdot T$, para parâmetro $0 < \alpha < 1$.

Simulated Annealing

Possíveis Critérios de parada no loop externo

- ▶ Seja T_{parada} uma temperatura final
- ▶ Condição atingida se $T_n \leq T_{parada}$

Possíveis Critérios de parada no loop interno

- ▶ Seja N um inteiro positivo
- ▶ Condição atingida se número de iterações seguidas sem melhorar solução atingiu N

Exemplo: TSP (D.S. Johnson e L.A. McGeoch'97)

S.A. com 2-OPT (Simulated Annealing com 2-OPT):

- ▶ Vizinhança 2-OPT
- ▶ Temperatura decrescendo de maneira geométrica ($\alpha = 0,95$)
- ▶ Temperaturas não calculadas de maneira exata (pelo exponencial), mas aproximadas por valores em tabela.
- ▶ Média de 10 execuções para $N = 100$ e de 5 execuções para N maiores.

$N =$	10^2	$10^{2.5}$	10^3
Apenas 2-OPT	4.5	4.8	4.9
S.A. com 2-OPT	5.2	4.1	4.1
S.A. com 2-OPT + Pós 2-OPT	3.4	3.7	4.0
S.A. sofisticado*	1.1	1.3	1.6

Excesso em relação ao limitante de Held-Karp

S.A. com 2-OPT + Pós 2-OPT = S.A. com 2-OPT aplicando 2-OPT novamente no tour gerado.

* Veja mais detalhes em Johnson & McGeoch'97.

Simulated Annealing

Exercícios faça algoritmos Simulated Annealing para os seguintes problemas:

1. Caixeiro Viajante: TSP
2. Corte de Peso Máximo: MaxCut
3. Satisfatibilidade de Peso Máximo: MaxSat
4. Subárvore de peso mínimo com exatamente k arestas

Introdução à Busca Tabu

- ▶ Baseado em Busca Local
- ▶ A cada iteração, procura nova solução vizinha, preferencialmente de custo melhor
- ▶ Sempre aceita uma nova solução que tiver custo melhor
- ▶ Cada solução é formada por elementos
- ▶ Tenta escapar de mínimos locais, proibindo alterações nos elementos afetados nas últimas k iterações
- ▶ Guarda a melhor solução encontrada durante sua execução

Busca Tabu

Notação e Termos:

k = Número das últimas iterações na **memória da Busca Tabu**

$T(k)$ = Lista de movimentos proibidos, **Lista Tabu**, considerando últimas k iterações.

$\mathcal{N}(S)$ = Conjunto de soluções vizinhas de S

- ▶ O **Critério de Aspiração** permite aceitar uma solução baseado em sua qualidade, mesmo que tal solução faça movimentos da lista tabu.

BUSCA TABU SIMPLIFICADA - MINIMIZAÇÃO

1. Seja S uma solução inicial
2. $S^+ \leftarrow S$ (mantém atualizado a melhor solução encontrada)
3. Enquanto não atingir condição de parada faça
4. Escolha soluções candidatas $V \subseteq \mathcal{N}(S)$
5. Enquanto $V \neq \emptyset$ faça
6. Pegue $S' \in V$ de peso mínimo e faça $V \leftarrow V \setminus \{S'\}$
7. Se $c(S') \leq c(S^+)$ então // (critério de aspiração)
8. $S^+ \leftarrow S'$, $S \leftarrow S'$ e $V \leftarrow \emptyset$
9. senão
10. se não é Tabu($S' \leftarrow S$) então
11. $S \leftarrow S'$ e $V \leftarrow \emptyset$
12. Devolva S^+

Possíveis implementações das subrotinas:

CONDIÇÕES DE PARADA

- ▶ Quando atingir limite de tempo de CPU
- ▶ Quando melhor solução não for atualizada por certo tempo

ESCOLHA DE SOLUÇÕES CANDIDATAS

- ▶ Escolha todas soluções vizinhas (se vizinhança for pequena)
- ▶ Escolha aleatoriamente um subconjunto dos vizinhos

Possíveis implementações das subrotinas:

CRITÉRIO-ASPIRAÇÃO

- ▶ No passo 7, note que mesmo uma solução com movimento Tabu, pode ser escolhida. Este é o critério de aspiração mais comum.

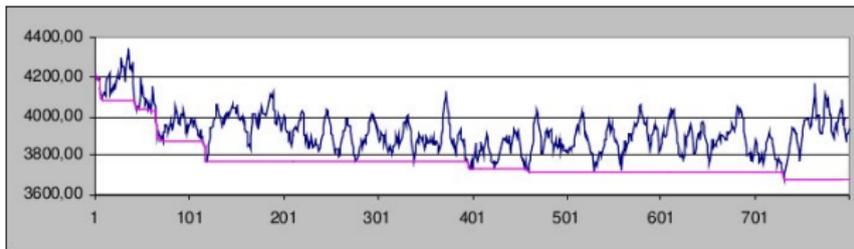
TABU($S' \leftarrow S$)

- ▶ Considere cada solução formada por elementos E
- ▶ Cada iteração do algoritmo de Busca Tabu representa um tempo
- ▶ Cada elemento $e \in E$ possui um marcador de tempo t_e (inicialmente $-\infty$)
- ▶ t_e é o último momento que e entrou ou saiu de uma solução
- ▶ Se estamos no momento t e for inserido/removido elemento e tempo t_e tal que $t - t_e \leq k$, então movimento é tabu.

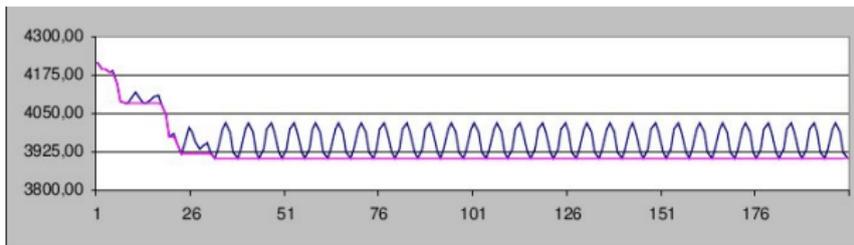
Valores adequados para k

Depende do problema, mas $k = 7$ é um bom ponto de partida

Exemplo de comportamento sem ciclo



Exemplo de comportamento com ciclo ($k = 3$)

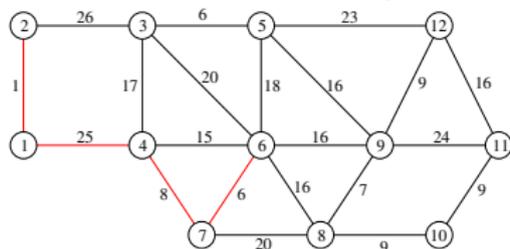


Exemplo: k -Árvore

k -Árvore: Dado um grafo $G = (V, E)$ com custo c_e em cada aresta $e \in E$, encontrar uma árvore com k vértices de peso mínimo.

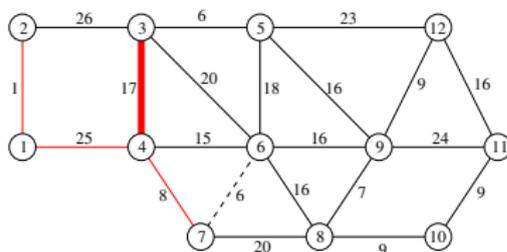
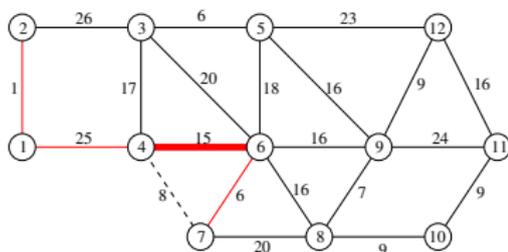
Teorema: O problema da k -Árvore é um problema NP-difícil.

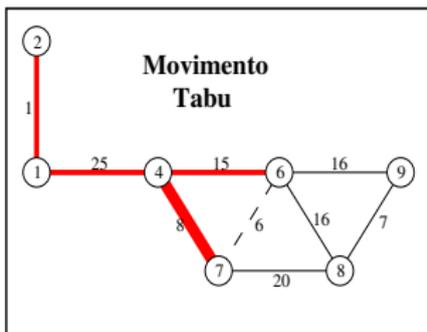
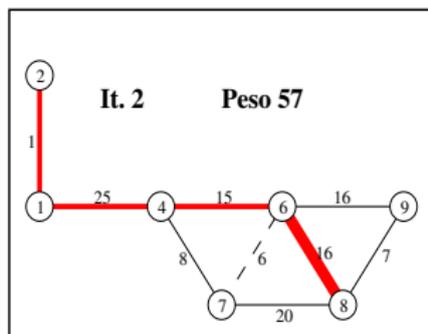
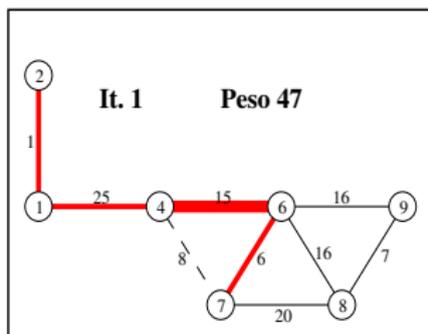
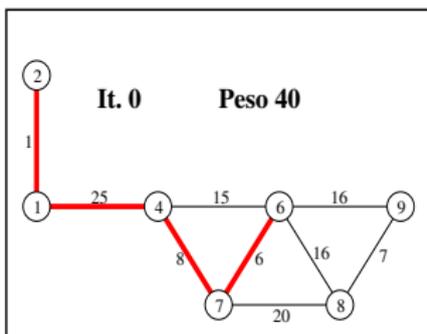
Movimentos:



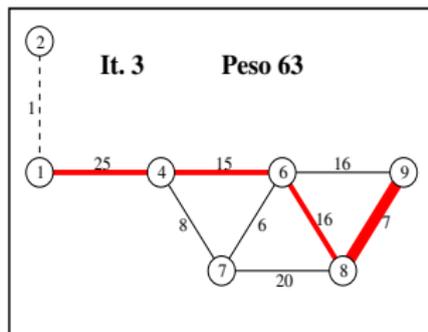
*Melhor movimento estatico
(mantem conjunto de vertices)*

Melhor movimento dinamico



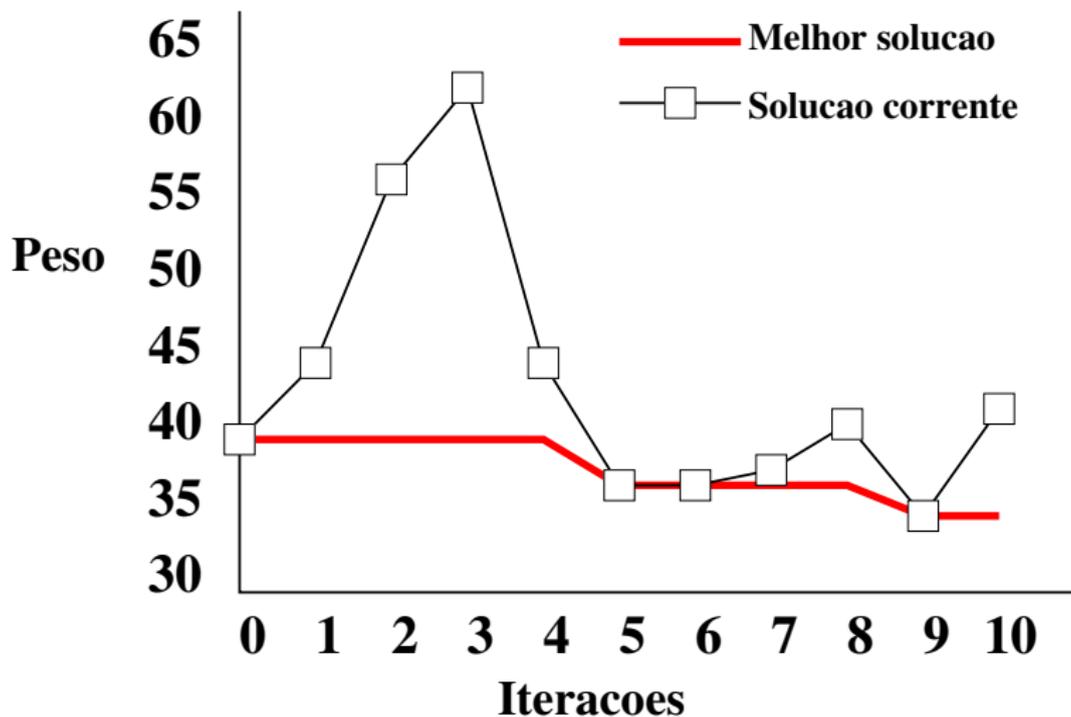


TABU



Arestas ficam na Lista Tabu por 2 iterações

Comportamento do algoritmo para 10 iterações com melhor movimento possível



Busca Tabu

Exercícios faça algoritmos Busca Tabu para os seguintes problemas:

1. Caixeiro Viajante: TSP
2. Corte de Peso Máximo: MaxCut
3. Satisfatibilidade de Peso Máximo: MaxSat