

Heurísticas Para Otimização Combinatória

Silvio Alexandre de Araujo

Slide 0

O que é Otimização Combinatória

- Problemas de otimização combinatória
 - * Dentre um conjunto finito (grande) de soluções, escolher a melhor.
 - * Esses problemas são modelados como problemas de maximizar (ou minimizar) uma função cujas variáveis devem obedecer certas restrições.

Slide 1

Exemplos Clássicos

- Problema da Mochila
 - Dados n objetos que posso armazenar em uma mochila, onde cada um tem um peso e uma utilidade, quanto de cada objeto devo escolher de tal modo que o peso total não seja maior que P e o somatório das utilidades seja o maior possível?
- Problema do Caixeiro Viajante
 - Dadas n cidades, encontrar um caminho passando por todas elas de maneira que o percurso total seja o menor possível.

Slide 2

Desafios Computacionais

- Resolver problemas de otimização combinatória pertencentes a classe NP-hard ou NP-completo com pouco esforço computacional (tempo de execução).
- Encontrar soluções ótimas, ou mesmo aproximadas, para esses tipos de problemas é um desafio nem sempre fácil de ser vencido
- Em muitos problemas práticos não há a necessidade de encontrar uma solução “teoricamente” ótima

Slide 3

Resolução de Problemas NP-difícil e NP-completo

- Algoritmos exatos não-polinomiais:
 - programação dinâmica , *branch-and-bound*, *branch-and-cut* , planos-de-corte, etc.
- Algoritmos aproximativos: encontram uma solução viável com um valor a uma distância máxima garantida do ótimo
- Algoritmos heurísticos:

Slide 4

Heurística

- Heurística é vem do grego *Heuriskein* (descobrir)
- Heurística é uma técnica que melhora a eficiência do processo de busca de soluções de um problema;
- As Heurísticas não são capazes de garantir a solução do problema;
- Os algoritmos **heurísticos** são algoritmos que não garantem encontrar a solução ótima de um problema, mas são capazes de retornar uma solução de qualidade em um tempo adequado para as necessidades da aplicação.
- O objetivo de uma heurística é tentar encontrar uma solução “boa” de maneira simples e rápida.

Slide 5

Heurística (classificação)

1. Métodos construtivos

- processo iterativo que inicia com uma solução vazia e adiciona um novo elemento a cada iteração até a obtenção de uma solução.

2. Métodos de decomposição

- consistem em dividir o problema em subproblemas menores, de modo que a resolução de todos os subproblemas possam compor uma solução para o problema maior.

3. Métodos de Redução

- identificam algumas características que presumidamente deverá possuir a solução ótima e simplifica o problema fixando-se o valor de tais variáveis

Slide 6

Heurística (classificação)

4. Manipulação do Modelo

- modificam o modelo de tal forma que ele fique mais fácil de resolver.
- Ex. relaxação linear, relaxação lagrangeana.

5. Métodos de Busca

- categoria a qual pertencem a maioria das meta-heurísticas
- inicia em uma solução (podendo ser obtida a partir de outra heurística) e caminha sobre as soluções vizinhas.

Slide 7

Heurística - Métodos Construtivos

Slide 8

Heurística - Método Construtivo

- Exemplos de algoritmos construtivos:

1. Problema do Caixeiro Viajante

- Vizinho mais próximo
- Inserção do Mais Distante

2. Problema da Mochila

- (H1) Escolher os itens por ordem decrescente da razão b_i/w_i , onde, b_i = valor e w_i = peso.
- (H2) Escolher os itens por ordem decrescente do valor (b_i).
- (H3) Escolher os itens por ordem crescente do valor (w_i).

Slide 9

Heurísticas Construtivas para Problema do Caixeiro Viajante (Traveling Salesman Problem) TSP

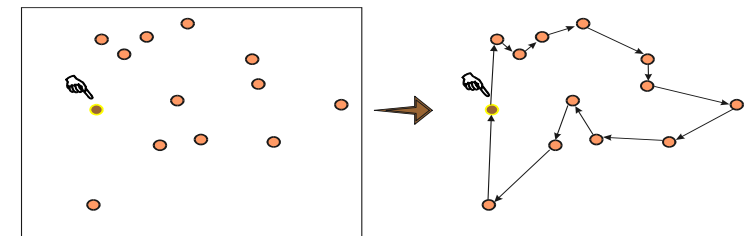
Silvio Alexandre de Araujo

<http://www.tsp.gatech.edu/index.html>

Slide 10

Problema do Caixeiro Viajante: Definição

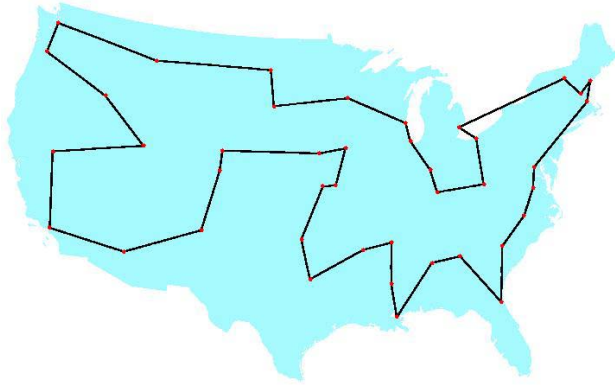
A um caixeiro viajante é informado um conjunto de cidades e um custo inteiro c_{ij} associado a cada cidade i e j deste conjunto, representando a distância de ir da cidade i à cidade j . O caixeiro deve partir de uma cidade inicial, passar por todas as demais uma única vez e retornar à cidade de partida. O problema consiste em fazer esta trajetória pelo menor caminho possível. No caso simétrico $c_{ij} = c_{ji} \forall i$ e j .



Slide 11

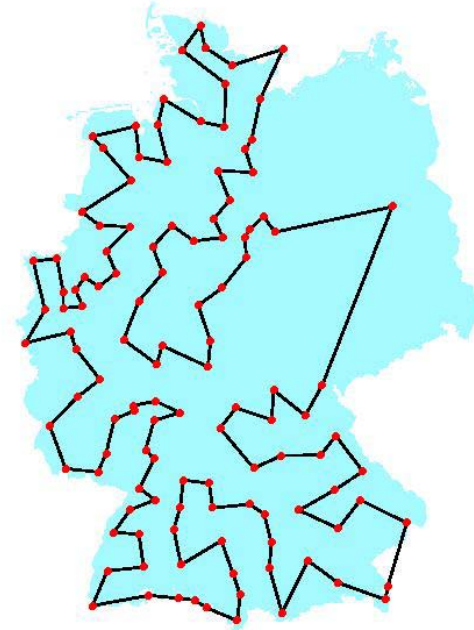
Problema do Caixeiro Viajante: Histórico

- 18xx: primeiros relatos sobre o problema;
- 192x: o problema foi definido;
- 194x: o problema foi popularizado e classificado como “difícil”;
- 1954: resolvido na otimalidade um problema de 42 cidades.



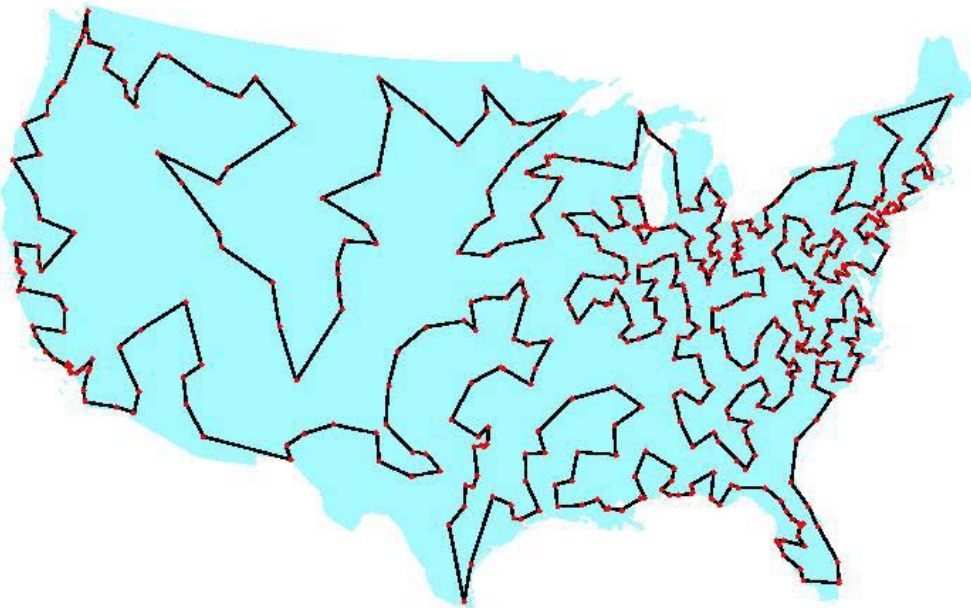
Slide 12

120 Cidades da Alemanha Ocidental (1977)

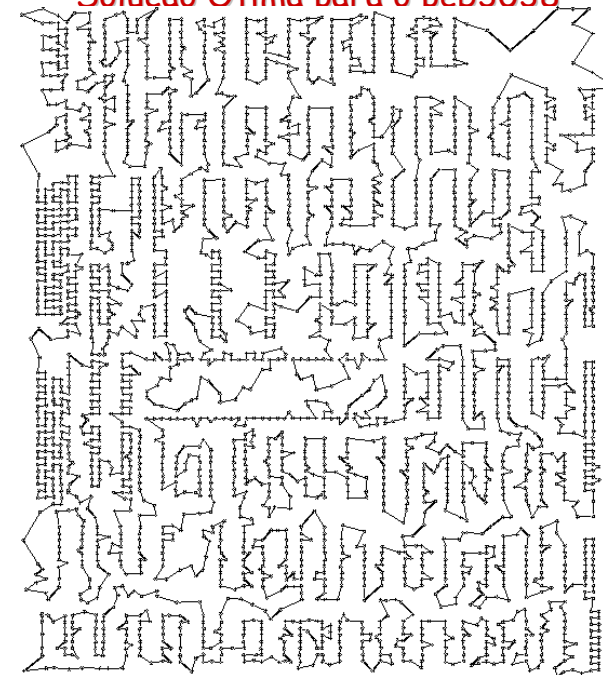


Slide 13

532 Cidades dos Estados Unidos (1987): att532



Solução Ótima para o pcb3038



Slide 15

Maior Instância Resolvida na Otimalidade (em 1998) :
usa13509



Problema do Caixeiro Viajante: Aplicações

- **Fabricação de placas de circuitos eletrônicos:** a fabricação de placas de circuitos eletrônicos é feita em diversas etapas:
- **Perfuração de placas circuito impresso (*Print Circuits Board*)**
- **Soldagem dos *chips* na placa**
- **Conexão entre os pinos**
- **Teste do circuito**

Slide 17

Problema do Caixeiro Viajante: Aplicações

- **Seqüenciamento de tarefas**
- Suponha que n tarefas devam ser processadas seqüencialmente em uma determinada máquina.
- O tempo de preparo da máquina para processar a tarefa j imediatamente após a tarefa i é designado por c_{ij} .
- O problema de encontrar uma seqüência de execução para as tarefas, de forma a minimizar o tempo total de processamento, pode ser modelado como um TSP

Slide 18

Problema do Caixeiro Viajante: Aplicações

- **Controle de robôs**
- Para fabricar alguma peça, um robô tem que realizar uma seqüência de operações.
- Para determinar uma seqüência para realizar as operações necessárias, a fim de minimizar o tempo de processamento (considerando que existem restrições de precedência) temos o problema de encontrar o caminho hamiltoniano mais curto

Slide 19

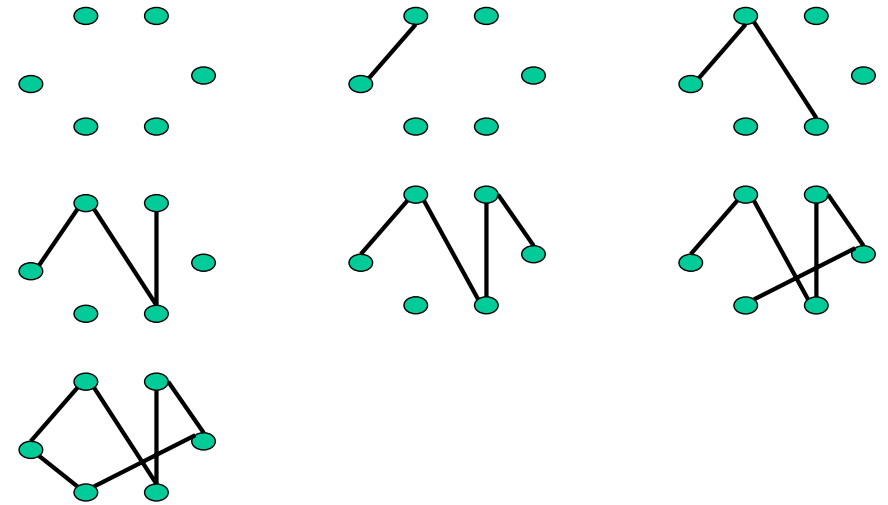
Problema do Caixeiro Viajante: Aplicações

- Roteamento de veículos como um problema de m -caixeiros

- clientes exigem certas quantidades de mercadorias e um fornecedor tem que satisfazer as demandas com uma frota de caminhões.
- tem-se o problema de designar clientes a caminhões e encontrar um escalonamento de entrega para cada caminhão de forma que a sua capacidade não seja excedida, bem como minimizar o custo total da viagem.
- se não há restrição de tempo ou se o número de caminhões é fixo (m), este problema pode ser resolvido como um TSP

Slide 20

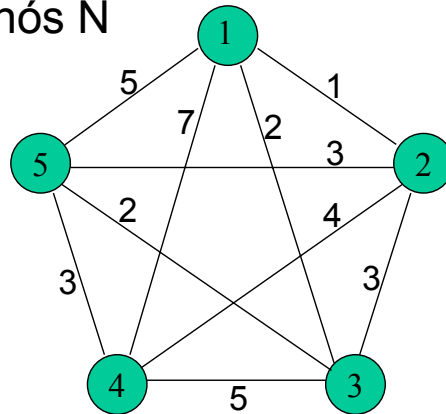
Algoritmos construtivos



Slide 21

Problema do caixeiro viajante

Matrix de distâncias c_{ij}
Conjunto de nós N
 $|N|=5$



Slide 22

Problema do caixeiro viajante

Algoritmo do vizinho mais próximo:

Escolher o nó inicial i e fazer $N \leftarrow N - \{i\}$.

Enquanto $N \neq \emptyset$ fazer:

Obter $j \in N$ tal que $c_{i,j} = \min_{k \in N} \{c_{i,k}\}$.

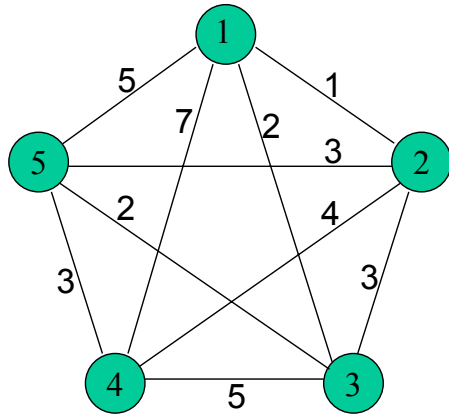
$N \leftarrow N - \{j\}$

$i \leftarrow j$

Fim-enquanto

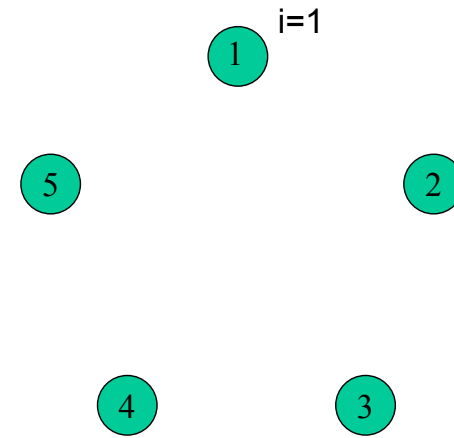
Slide 23

Problema do caixeiro viajante



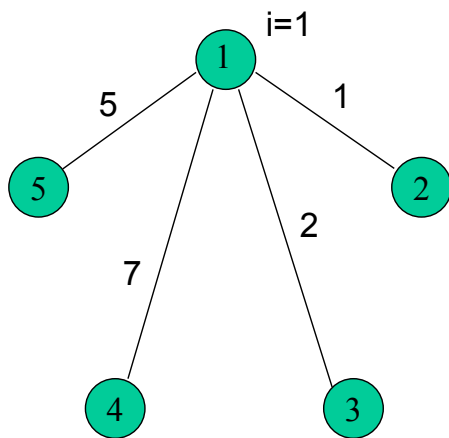
Slide 24

Problema do caixeiro viajante



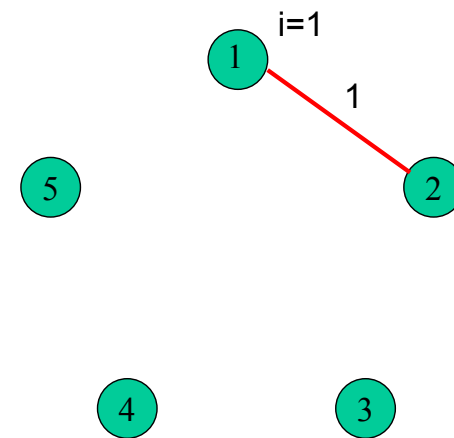
Slide 25

Problema do caixeiro viajante



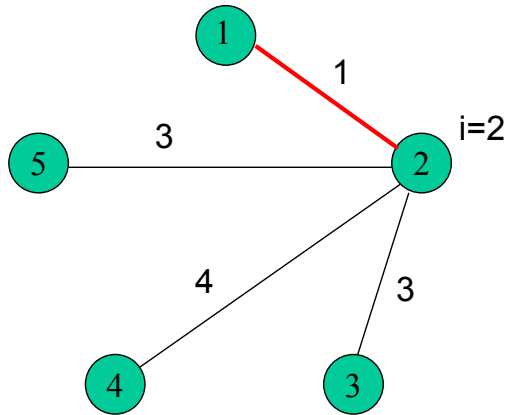
Slide 26

Problema do caixeiro viajante



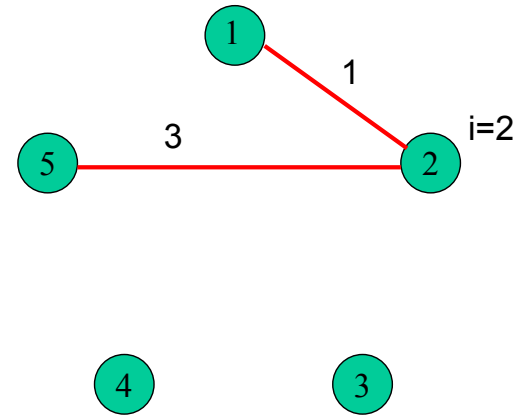
Slide 27

Problema do caixeiro viajante



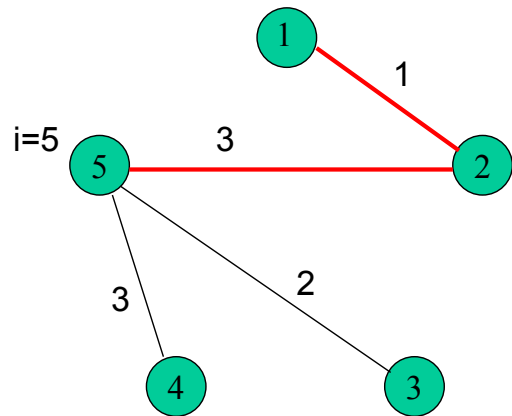
Slide 28

Problema do caixeiro viajante



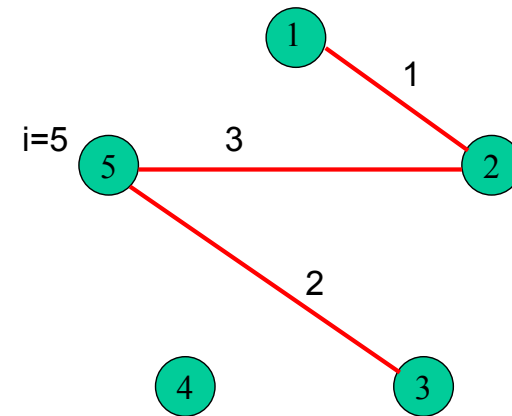
Slide 29

Problema do caixeiro viajante



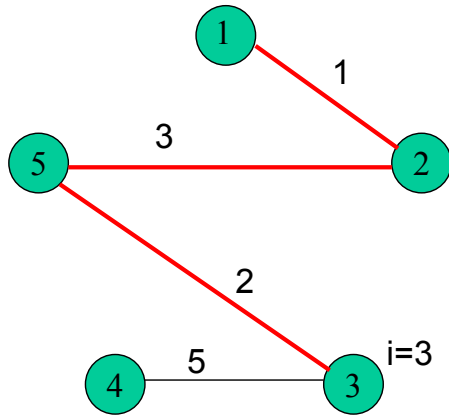
Slide 30

Problema do caixeiro viajante



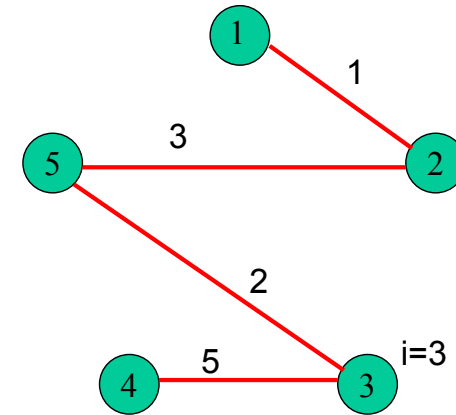
Slide 31

Problema do caixeiro viajante



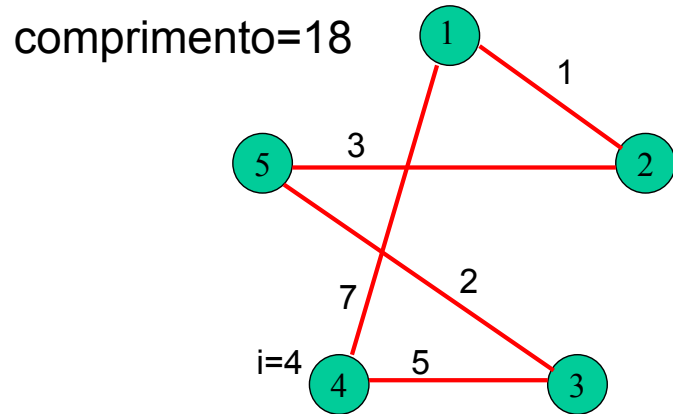
Slide 32

Problema do caixeiro viajante



Slide 33

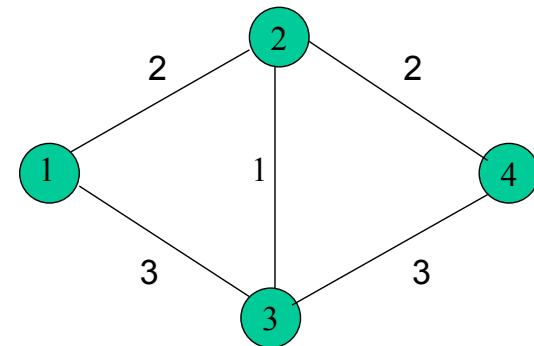
Problema do caixeiro viajante



Slide 34

Problema do caixeiro viajante

- Algoritmos construtivos simples podem falhar mesmo para casos muito simples:



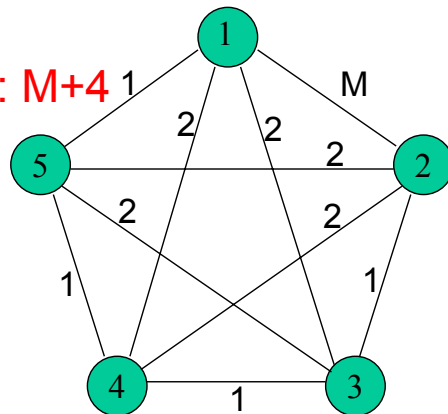
Slide 35

Problema do caixeiro viajante

- Podem encontrar soluções arbitrariamente ruins:

Heurística (1-5-4-3-2-1): $M+4$

Ótimo (1-5-2-3-4-1): 7



A solução ótima é encontrada se o algoritmo começa do nó 5.

Slide 36

Problema do caixeiro viajante

- Extensões e melhorias simples:
 - * Repetir a aplicação do algoritmo a partir de cada nó do grafo e selecionar a melhor solução obtida.
 - Melhores soluções, mas tempo de processamento multiplicado por n .
 - * A cada iteração selecionar a aresta mais curta a partir de alguma das extremidades em aberto do circuito, e não apenas a partir da última extremidade inserida: **solução de comprimento 15** (tempos multiplicados por dois).
 - * Critérios mais elaborados para (1) seleção do novo nó incorporado ao circuito a cada iteração e para (2) seleção da posição onde ele entra no circuito: **algoritmo baseado no crescimento de um circuito até completá-lo**.

Slide 37

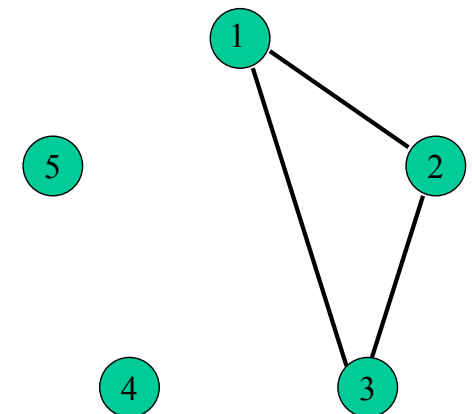
Problema do caixeiro viajante

- Escolha do novo nó a ser incorporado ao circuito a cada iteração:
 - * Selecionar o nó k fora do circuito parcial corrente, cuja aresta de menor comprimento que o liga a este circuito parcial corrente é **mínima** → algoritmo de inserção mais próxima
 - * Selecionar o nó k fora do circuito parcial corrente, cuja aresta de menor comprimento que o liga a este circuito parcial corrente é **máxima** → algoritmo de inserção mais afastada

Slide 38

Problema do caixeiro viajante

- Escolha do novo nó a ser incorporado ao circuito a cada iteração pela inserção mais próxima
- Seleção do nó:



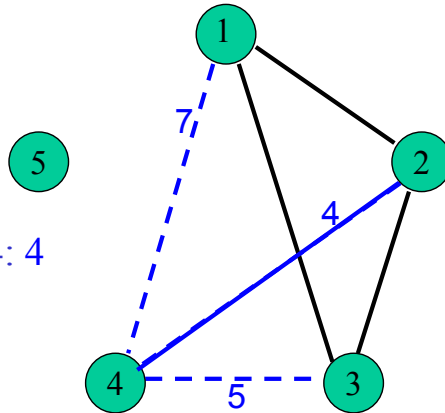
Slide 39

Problema do caixeiro viajante

- Escolha do novo nó a ser incorporado ao circuito a cada iteração pela inserção mais próxima

- Seleção do nó:

* Distância mínima do nó 4: 4



Slide 40

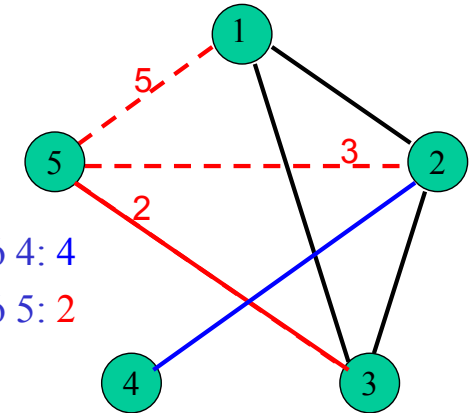
Problema do caixeiro viajante

- Escolha do novo nó a ser incorporado ao circuito a cada iteração pela inserção mais próxima

- Seleção do nó:

* Distância mínima do nó 4: 4

* Distância mínima do nó 5: 2



Slide 41

Problema do caixeiro viajante

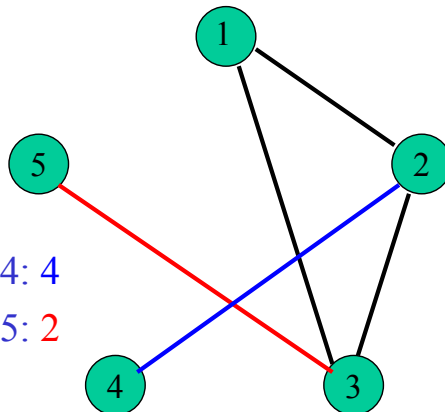
- Escolha do novo nó a ser incorporado ao circuito a cada iteração pela inserção mais próxima

- Seleção do nó:

* Distância mínima do nó 4: 4

* Distância mínima do nó 5: 2

* Nó selecionado: 5

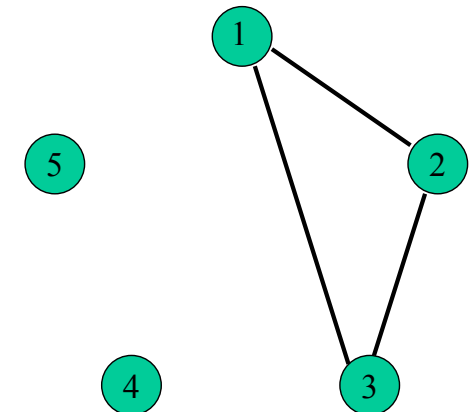


Slide 42

Problema do caixeiro viajante

- Escolha do novo nó a ser incorporado ao circuito a cada iteração pela inserção mais afastada

- Seleção do nó:



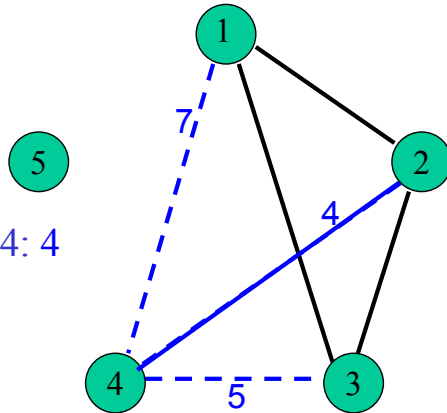
Slide 43

Problema do caixeiro viajante

- Escolha do novo nó a ser incorporado ao circuito a cada iteração pela inserção mais afastada

- Seleção do nó:

- * Distância mínima do nó 4: 4



Slide 44

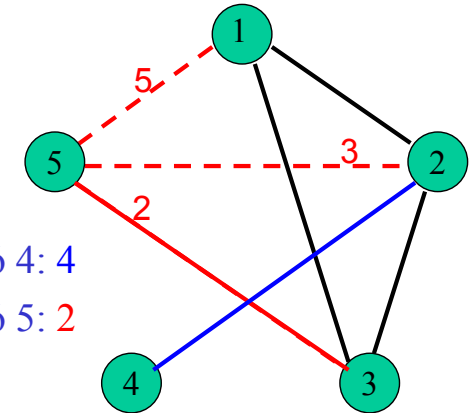
Problema do caixeiro viajante

- Escolha do novo nó a ser incorporado ao circuito a cada iteração pela inserção mais afastada

- Seleção do nó:

- * Distância mínima do nó 4: 4

- * Distância mínima do nó 5: 2



Slide 45

Problema do caixeiro viajante

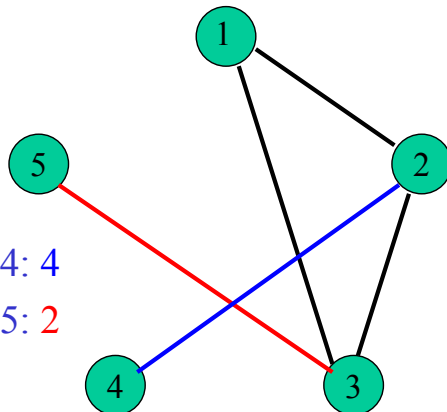
- Escolha do novo nó a ser incorporado ao circuito a cada iteração pela inserção mais afastada

- Seleção do nó:

- * Distância mínima do nó 4: 4

- * Distância mínima do nó 5: 2

- * Nó selecionado: 4



Slide 46

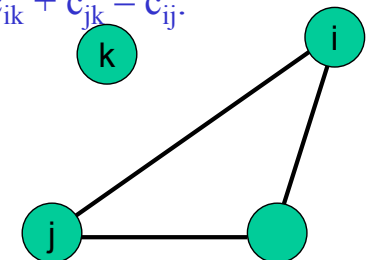
Problema do caixeiro viajante

- Escolha da posição onde o nó selecionado entra no circuito:

- * Suponha a entrada do nó k entre o nó i e o nó j.

- * Devem ser inseridas as arestas (i,k) e (j,k) no circuito parcial corrente, substituindo a aresta (i,j).

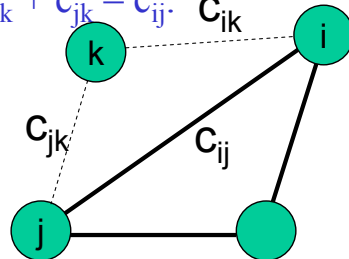
- * A variação no comprimento do circuito parcial corrente é dada por $\Delta_{ij}(k) = c_{ik} + c_{jk} - c_{ij}$.



Slide 47

Problema do caixeiro viajante

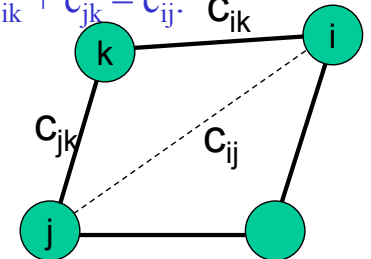
- Escolha da posição onde o nó selecionado entra no circuito:
 - * Suponha a entrada do nó k entre o nó i e o nó j.
 - * Devem ser inseridas as arestas (i,k) e (j,k) no circuito parcial corrente, substituindo a aresta (i,j).
 - * A variação no comprimento do circuito parcial corrente é dada por $\Delta_{ij}(k) = c_{ik} + c_{jk} - c_{ij}$.



Slide 48

Problema do caixeiro viajante

- Escolha da posição onde o nó selecionado entra no circuito:
 - * Suponha a entrada do nó k entre o nó i e o nó j.
 - * Devem ser inseridas as arestas (i,k) e (j,k) no circuito parcial corrente, substituindo a aresta (i,j).
 - * A variação no comprimento do circuito parcial corrente é dada por $\Delta_{ij}(k) = c_{ik} + c_{jk} - c_{ij}$.

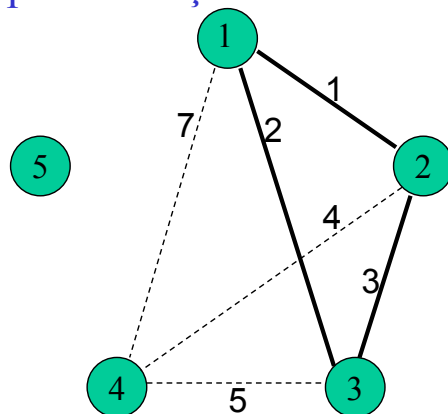


Slide 49

Problema do caixeiro viajante

- Escolha da posição onde o nó selecionado entra no circuito parcial corrente
- Exemplo: nó 4 escolhido para inserção

$$\begin{aligned}\Delta_{12}(4) &= 7+4-1 = 10 \\ \Delta_{13}(4) &= 7+5-2 = 10 \\ \Delta_{23}(4) &= 5+4-3 = 6\end{aligned}$$



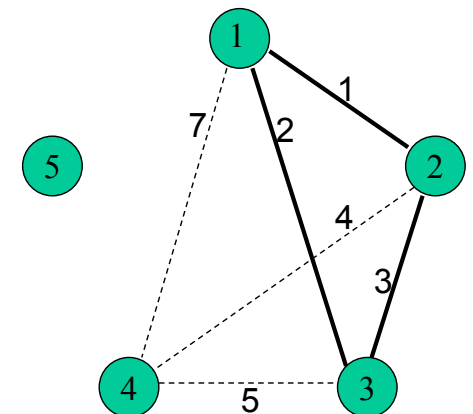
Slide 50

Problema do caixeiro viajante

- Escolha da posição onde o nó selecionado entra no circuito parcial corrente
- Exemplo: nó 4 escolhido para inserção

$$\begin{aligned}\Delta_{12}(4) &= 7+4-1 = 10 \\ \Delta_{13}(4) &= 7+5-2 = 10 \\ \Delta_{23}(4) &= 5+4-3 = 6\end{aligned}$$

- Inserção mais próxima: entre os nós 2 e 3



Slide 51

Problema do caixeiro viajante

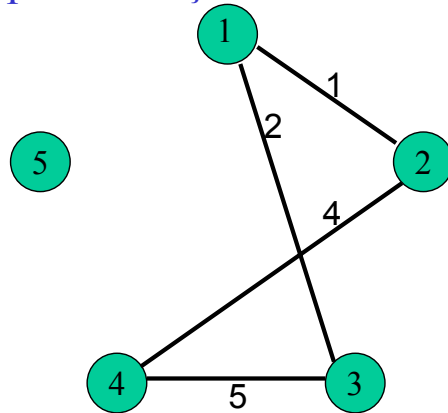
- Escolha da posição onde o nó selecionado entra no circuito parcial corrente
- Exemplo: nó 4 escolhido para inserção

$$\Delta_{12}(4) = 7+4-1 = 10$$

$$\Delta_{13}(4) = 7+5-2 = 10$$

$$\Delta_{23}(4) = 5+4-3 = 6$$

- Inserção mais próxima: entre os nós 2 e 3



Slide 52

Problema do caixeiro viajante

- Algoritmo de inserção mais próxima:

Escolher o nó inicial i , inicializar um circuito apenas com o nó i e fazer $N \leftarrow N - \{i\}$.

Enquanto $N \neq \emptyset$ fazer:

Encontrar o vértice k fora do circuito corrente cuja aresta de menor comprimento que o liga a ele é mínima.

Encontrar o par de arestas (i,k) e (j,k) que ligam o vértice k ao ciclo minimizando $\Delta_{ij}(k) = c_{ik} + c_{jk} - c_{ij}$.

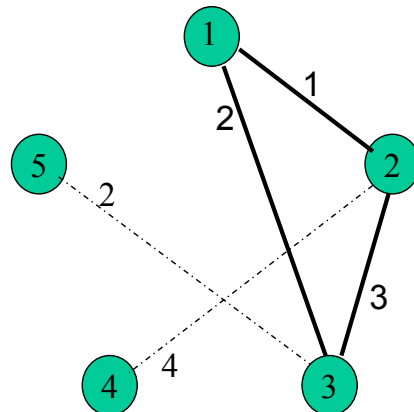
Inserir as arestas (i,k) e (j,k) e retirar a aresta (i,j) .

Fazer $N \leftarrow N - \{k\}$.

Fim-enquanto

Slide 53

Problema do caixeiro viajante



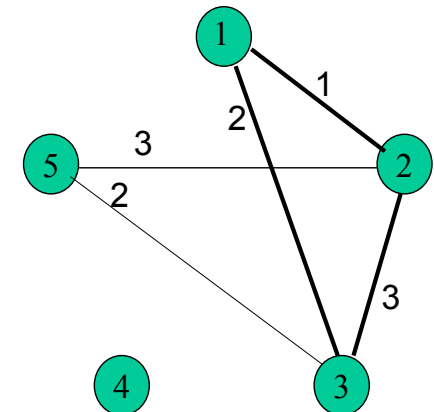
Slide 54

Problema do caixeiro viajante

$$\Delta_{12}(5) = 5+3-1 = 7$$

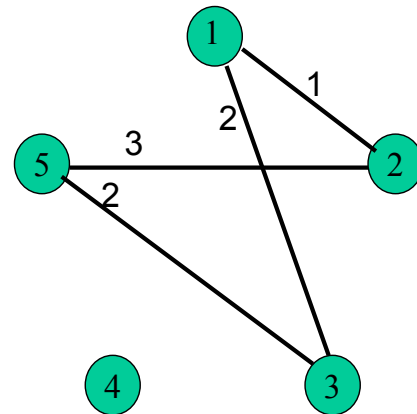
$$\Delta_{23}(5) = 2+3-3 = 2$$

$$\Delta_{13}(5) = 2+5-2 = 5$$



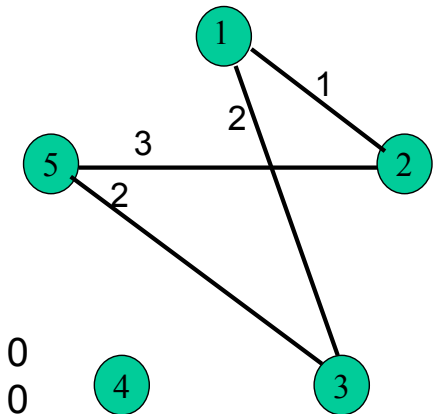
Slide 55

Problema do caixeiro viajante



Slide 56

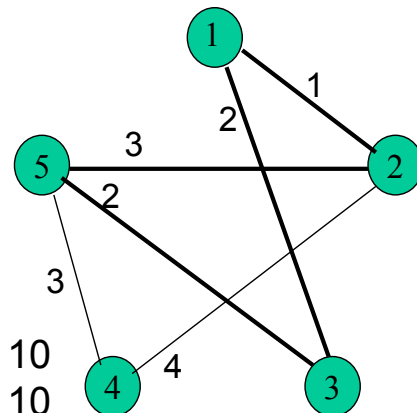
Problema do caixeiro viajante



$$\begin{aligned}\Delta_{12}(4) &= 7+4-1 = 10 \\ \Delta_{13}(4) &= 7+5-2 = 10 \\ \Delta_{35}(4) &= 5+3-2 = 6 \\ \Delta_{25}(4) &= 4+3-3 = 4\end{aligned}$$

Slide 57

Problema do caixeiro viajante

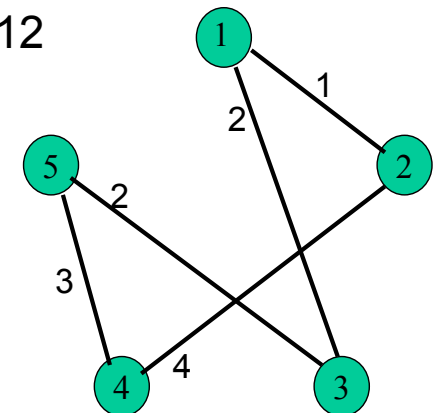


$$\begin{aligned}\Delta_{12}(4) &= 7+4-1 = 10 \\ \Delta_{13}(4) &= 7+5-2 = 10 \\ \Delta_{35}(4) &= 5+3-2 = 6 \\ \Delta_{25}(4) &= 4+3-3 = 4\end{aligned}$$

Slide 58

Problema do caixeiro viajante

comprimento = 12



Slide 59

Problema do caixeiro viajante

- Comparação: na prática, o método de inserção mais afastada alcança melhores resultados do que o de inserção mais próxima.
- Melhoria simples: método de inserção mais barata
 - * Por que separar em dois passos (1) a seleção do novo nó incorporado ao circuito a cada iteração e (2) a seleção da posição onde ele entra no circuito?
 - * Fazer a escolha da melhor combinação em conjunto.
 - * Melhores soluções, mas tempos de processamento maiores (cerca de n vezes maiores).

Slide 60

Problema do caixeiro viajante

- Algoritmo de inserção mais barata:

Escolher o nó inicial i , inicializar um circuito apenas com o nó i e fazer $N \leftarrow N - \{i\}$.

Enquanto $N \neq \emptyset$ fazer:

Encontrar o vértice k fora do circuito corrente e o par de arestas (i,k) e (j,k) que ligam o vértice k ao ciclo minimizando $\Delta_{ij}(k) = c_{ik} + c_{jk} - c_{ij}$.

Inserir as arestas (i,k) e (j,k) e retirar a aresta (i,j) .

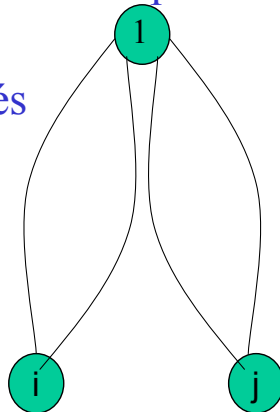
Fazer $N \leftarrow N - \{k\}$.

Fim-enquanto

Slide 61

Problema do caixeiro viajante

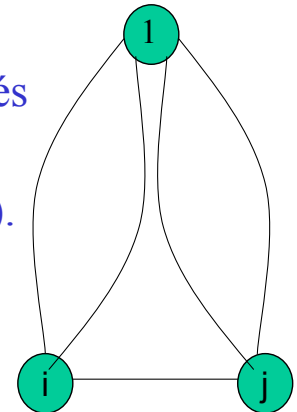
- Outra idéia diferente: considerar a fusão de subcircuitos
- Considerar dois subcircuitos passando pelo nó 1 e pelos nós i e j .
- Conectá-los diretamente através da aresta (i,j) .



Slide 62

Problema do caixeiro viajante

- Outra idéia diferente: considerar a fusão de subcircuitos
- Considerar dois subcircuitos passando pelo nó 1 e pelos nós i e j .
- Conectá-los diretamente através da aresta (i,j) .
- Remover as arestas $(1,i)$ e $(1,j)$.

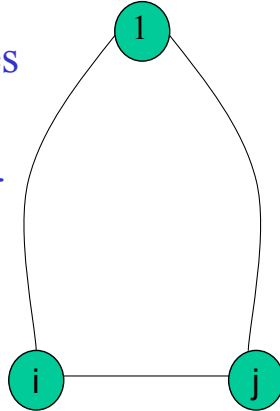


Slide 63

Problema do caixeiro viajante

- Outra idéia diferente: considerar a fusão de subcircuitos
- Considerar dois subcircuitos passando pelo nó 1 e pelos nós i e j.
- Conectá-los diretamente através da aresta (i,j).
- Remover as arestas (1,i) e (1,j).
- Economia realizada:

$$s_{ij} = c_{1i} + c_{1j} - c_{ij}$$



Slide 64

Problema do caixeiro viajante

- Algoritmo das economias:

Escolher um nó inicial i (e.g., $i = 1$).

Construir subcircuitos de comprimento 2 envolvendo o nó inicial (e.g., $i = 1$) e cada um dos demais nós de N.

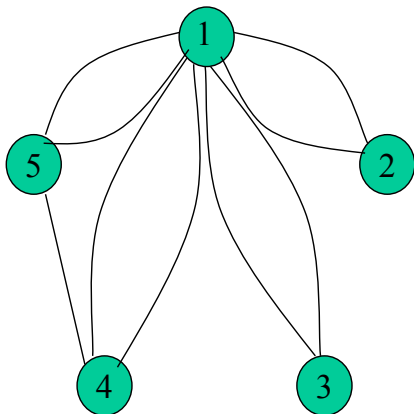
Calcular as economias $s_{ij} = c_{1i} + c_{1j} - c_{ij}$ obtidas pela fusão dos subcircuitos contendo i e j e ordená-las em ordem decrescente.

Percorrer a lista de economias e fundir os subcircuitos possíveis: a cada iteração, maximizar a distância economizada sobre a solução anterior, combinando-se dois subcircuitos e substituindo-os por uma nova aresta.

Slide 65

Problema do caixeiro viajante

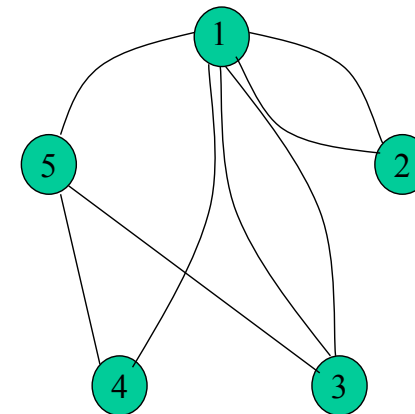
$$\begin{aligned} s_{45} &= 9 \\ s_{35} &= 5 \\ s_{34} &= 4 \\ s_{24} &= 4 \\ s_{25} &= 3 \\ s_{23} &= 0 \end{aligned}$$



Slide 66

Problema do caixeiro viajante

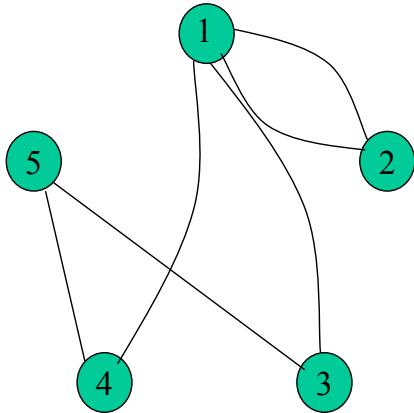
$$\begin{aligned} s_{45} &= 9 \\ s_{35} &= 5 \\ s_{34} &= 4 \\ s_{24} &= 4 \\ s_{25} &= 3 \\ s_{23} &= 0 \end{aligned}$$



Slide 67

Problema do caixeiro viajante

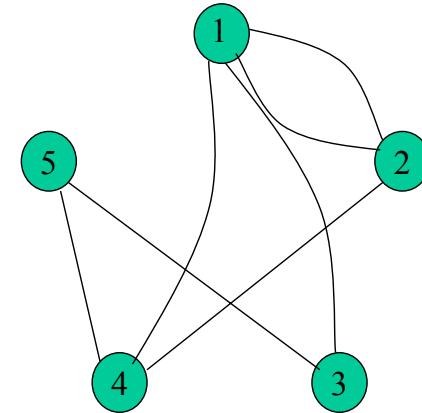
$$\begin{aligned}s_{45} &= 9 \\ s_{35} &= 5 \\ s_{34} &= 4 \\ s_{24} &= 4 \\ s_{25} &= 3 \\ s_{23} &= 0\end{aligned}$$



Slide 68

Problema do caixeiro viajante

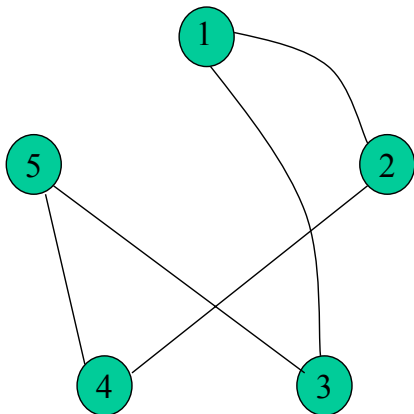
$$\begin{aligned}s_{45} &= 9 \\ s_{35} &= 5 \\ s_{34} &= 4 \\ s_{24} &= 4 \\ s_{25} &= 3 \\ s_{23} &= 0\end{aligned}$$



Slide 69

Problema do caixeiro viajante

$$\begin{aligned}s_{45} &= 9 \\ s_{35} &= 5 \\ s_{34} &= 4 \\ s_{24} &= 4 \\ s_{25} &= 3 \\ s_{23} &= 0\end{aligned}$$



comprimento = 12

Slide 70

Exercício

- Implementar diferentes heurísticas de construção para resolução do Problema do Caixeiro Viajante

Slide 71

Comentários

- Descrição de instâncias e soluções ótimas para o problema do caixeiro viajante:
<http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>
- Estudo comparativo de heurísticas para o problema do caixeiro viajante:
<http://www.research.att.com/~dsj/chtsp/>
- Em particular, página com gráficos comparativos:
<http://www.research.att.com/~dsj/chtsp/testform2.html>
- Outra Página:
<http://www.tsp.gatech.edu/index.html>

Slide 72

Comentários

- Referências:
 - Lawler, Lenstra, Rinnooy Kan e Shmoys (eds.), “The traveling salesman problem”, 1985
 - David S. Johnson, Lyle A. McGeoch, “The Traveling Salesman Problem: A Case Study in Local Optimization” (1995)

Slide 73

Heurística - Métodos de Busca

Slide 74

Bibliografia

- A. Diaz, F. Glover, H. M. Ghaziri, J. L. González, M. Laguna, P. Moscato e F. T. Tseng, *Optimización Heurística Y Redes Neurolales*, Editorial Paraninfo, Espanha, 1996.
- Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-verlag, 1992.
- C. R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell, 1993.
- F. Glover e M. Laguna, *Tabu Search*, Kluwer Academic Publishers, USA, 1997.
- V. J. Rayward-smith, I. H. Osman, C. R. Reeves e G. D. Smith, *Modern Heuristic Search Methods*, Wiley, Inglaterra, 1996.

Slide 75

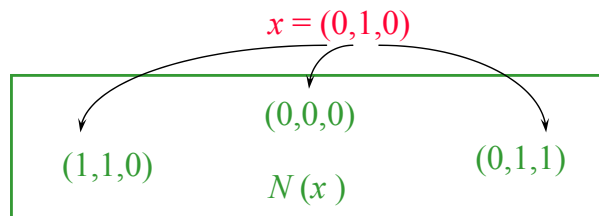
Conceitos básicos

- Espaço de soluções

- * O conjunto de todas as soluções (viáveis) possíveis (satisfazendo as restrições do problema)

- Vizinhaça

- * Dada uma solução x , os elementos da vizinhaça $N(x)$ de x são aquelas soluções y que pode ser obtida aplicando uma perturbação elementar sobre x .
- * Exemplo: Considere $x = (0,1,0)$ e a vizinhaça 1-flip de um vetor 0/1.



Slide 76

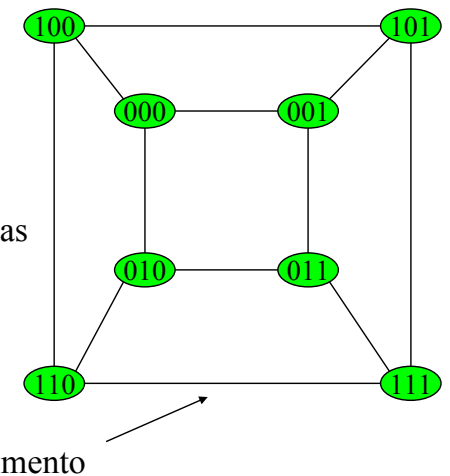
Espaço de busca

- Movimento

- * É a transição de uma solução para outra solução vizinha.

- Espaço de busca

- * O conjunto das soluções obtidas por meio de uma vizinhaça.



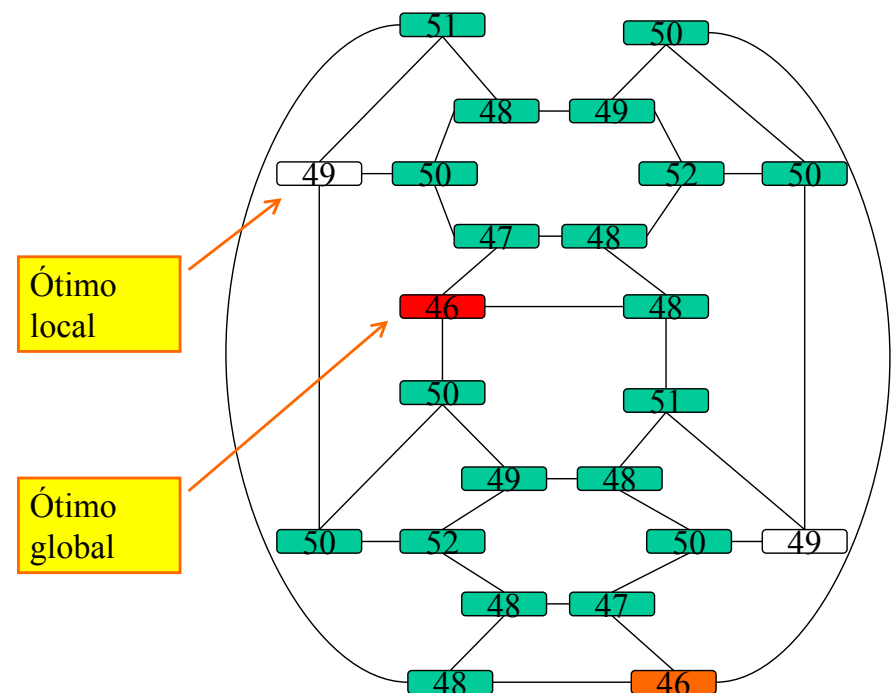
Slide 77

- Ótimo local

- * é uma solução tão boa ou melhor do qualquer das soluções vizinhas

- Ótimo global (solução ótima).

- * É a melhor solução dentre todos os ótimos locais.



Slide 78

Slide 79

Heurística - Busca Local

- Algoritmos de busca local são construídos como uma forma de exploração do espaço de busca.
- Partida: solução inicial obtida através de um método construtivo
- Iteração: melhoria sucessiva da solução corrente através de uma busca na sua vizinhança
- Parada: primeiro ótimo local encontrado, ou seja, não existe solução vizinha melhor.

Slide 80

Representação de soluções

- Os métodos de busca local são dependentes da forma adotada para representar as soluções, pois a vizinhança de uma solução é obtida a partir da sua representação.
- Representação de uma solução: indicar quais elementos estão presentes e quais não estão.
- As formas mais adotadas são:
 - * Vetores de Pertinência
 - * Combinações
 - * Permutações

Slide 81

Representação de soluções

• Vetores de pertinência

Vetores onde o mapeamento de cada elemento da solução é feito para cada uma das posições (ou dimensões) do vetor. A presença de um determinado elemento na resposta é representada pelo 1 na posição correspondente. Sua ausência é representada pelo 0 (ou vice-versa)

• Combinações

Conjuntos contendo exatamente os elementos presentes na resposta (ou exatamente os elementos ausentes).

• Permutações

Conjunto ordinal (contendo todos os elementos) indicando a ordem com que estes aparecem na solução. (ex.: lista de cidades visitadas para o problema do caixeiro viajante).

Slide 82

Representação de soluções

- Vetor de pertinência para o problema da mochila 0-1: n itens, vetor 0-1 com n posições, $x_j = 1$ se o item j é selecionado, $x_j = 0$ caso contrário.

$$S = \begin{array}{c} 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \\ \boxed{1} \boxed{1} \boxed{0} \boxed{0} \boxed{0} \boxed{1} \boxed{0} \boxed{1} \boxed{1} \boxed{0} \end{array}$$

$$S[i] = \begin{cases} 1 & \text{se o objeto } i \text{ está na resposta} \\ 0 & \text{caso contrário} \end{cases}$$

$$\sum_{i=1}^{10} S[i]W_i \leq C$$

Slide 83

Representação de soluções

- Combinação para o problema da mochila:

Uma solução S é formada pelo conjunto de itens selecionados.

$$S = \{1, 2, 6, 8, 9\}$$

$$\sum_{i \in S} W_i \leq C$$

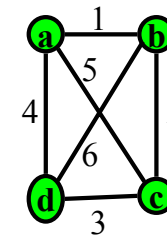
Slide 84

Representação de soluções

Vetor de pertinência para o PCV:

Solução é um vetor de n = número de arestas posições

$v_e = 1$, se a aresta e pertence a solução
0, caso contrário.



Soluções viáveis:

$(1, 1, 1, 1, 0, 0)$, $(1, 0, 1, 0, 1, 1)$, $(0, 1, 0, 1, 1, 1)$

Slide 85

Representação de soluções

Permutação para o PCV:

Cada solução é representada pela ordem em que os vértices são visitados, isto é, como uma permutação circular dos n vértices (já que o primeiro vértice é arbitrário)

(a)bcd	(b)bdc
(c)cbd	(d)cdb
(e)dbc	(f)dc b

Slide 86

Vizinhança

- Problema combinatório:

$f(s^*) = \text{mínimo } \{ f(s) : s \in S \}$

S é um conjunto discreto de soluções

- Vizinhança: elemento que introduz a noção de proximidade entre as soluções em S .
- Uma **vizinhança** é um mapeamento que leva as soluções de S em um subconjunto deste mesmo conjunto de soluções.

Slide 87

Vizinhança

$N(s) = \{s_1, s_2, \dots, s_k\}$ soluções vizinhas de s

- Boas vizinhanças permitem representar de forma compacta e eficiente o conjunto de soluções vizinhas de qualquer solução s .
- Espaço de busca: definido pelo conjunto de soluções S e por uma vizinhança N

Slide 88

Vizinhança

Exemplo de vizinhanças no espaço de permutações:

Solução $\pi = (\pi_1, \dots, \pi_{i-1}, \pi_i, \pi_{i+1}, \dots, \pi_j, \dots, \pi_n)$

$N1(\pi) = \{(\pi_1, \dots, \pi_{i+1}, \pi_i, \dots, \pi_n) : i=1, \dots, n-1\}$

Vizinhos de $(1,2,3,4) = \{(2,1,3,4), (1,3,2,4), (1,2,4,3)\}$

$N2(\pi) = \{(\pi_1, \dots, \pi_j, \dots, \pi_i, \dots, \pi_n) : i=1, \dots, n-1; j=i+1, \dots, n\}$

Vizinhos de $(1,2,3,4) = \{(2,1,3,4), (1,3,2,4), (1,2,4,3), (4,2,3,1), (2,3,1,4), (1,4,3,2)\}$

Slide 89

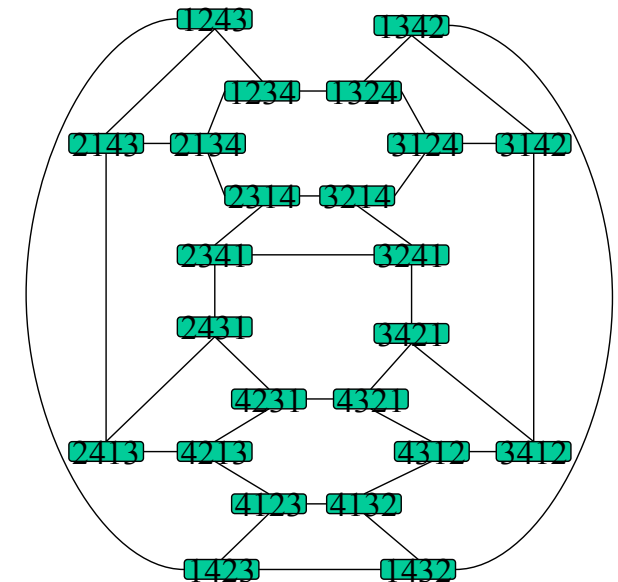
Vizinhança

- O espaço de busca pode ser visto como um grafo onde os vértices são as soluções e existem arestas entre pares de vértices associados a soluções vizinhas.
- Este espaço pode ser visto como uma superfície com vales e cumes definidos pelo valor e pela proximidade (vizinhança) das soluções.
- Um caminho no espaço de busca consiste numa sequência de soluções, onde duas soluções consecutivas quaisquer são vizinhas.

Slide 90

Vizinhança/Espaço de Busca

Exemplo 1: espaço de busca para a vizinhança $N1$ sobre um conjunto solução de 4 dígitos.

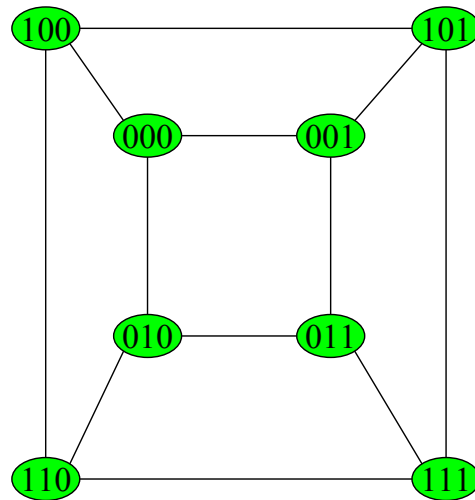


Slide 91

Vizinhança/Espaço de Busca

Exemplo 2: vetores de pertinência 0-1 com a vizinhança $N3$ definida como:

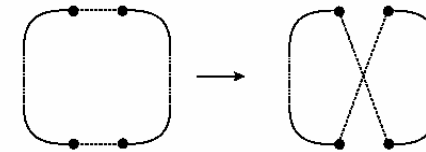
$v = (v_1, \dots, v_i, \dots, v_n)$
 $N3(v) = \{(v_1, \dots, 1-v_i, \dots, v_n) : i=1, \dots, n\}$
 Vizinhos de $(1,0,1) = \{(0,0,1), (1,1,1), (1,0,0)\}$



Slide 92

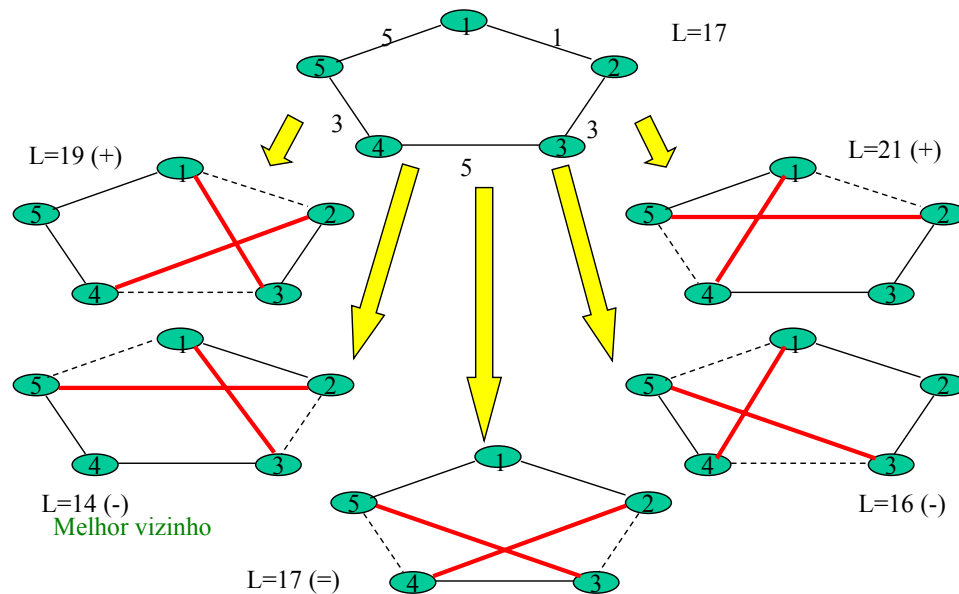
Vizinhança - mais exemplos

- K-opt: uma heurística de busca local aplicada no problema do caixeiro viajante. A vizinhança $N(s)$ é formada por ciclos obtidos pela substituição de k arestas do ciclo corrente S por k arestas que não estão no ciclo.
- Ex: 2-opt



Slide 93

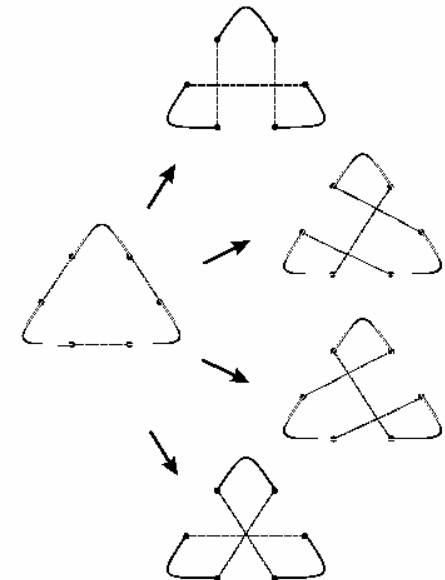
Busca local para o PCV



Slide 94

Vizinhança - mais exemplos

Ex: vizinhança 3-opt para o PCV.



Slide 95

Busca Local

Questões fundamentais:

1. Representação da solução
2. Definição da vizinhança
3. Estratégia de busca na vizinhança
4. Complexidade de cada iteração:
 - a) Proporcional ao tamanho da vizinhança
 - b) Eficiência depende da forma como é calculada a variação da função objetivo para cada solução vizinha: algoritmos eficientes são capazes de recalculas as variações de modo a atualizá-las quando a solução corrente se modifica, evitando cálculos repetitivos e desnecessários da função objetivo.

Slide 96

Busca Local: problema de minimização

inicie com uma solução S

faça

$melhora \leftarrow 0$

$S' \leftarrow$ seleciona

$\Delta\text{custo} \leftarrow \text{custo}(S') - \text{custo}(S)$

se $\Delta\text{custo} < 0$ então

$S \leftarrow S'$

$melhora \leftarrow 1$

enquanto ($melhora$)

Slide 97

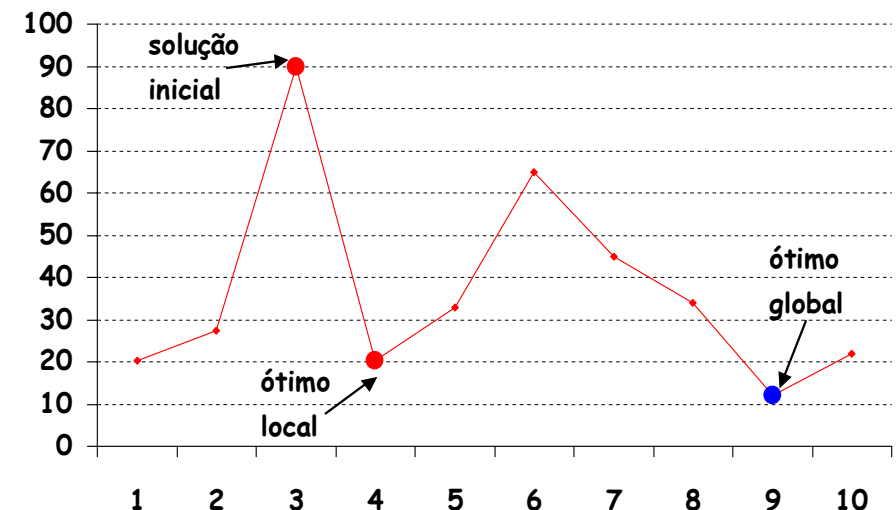
Busca Local

- seleciona pode ser:

- * (H1) - Analise todas as soluções vizinhas e escolha a que leve a menor Δcusto
 - analisar todas as soluções pode ser computacionalmente muito caro
- * (H2) - Analise as soluções vizinhas e escolha a primeira que leve a $\Delta\text{custo} < 0$.

Slide 98

Busca Local



Slide 99

Busca Local - Prob. da Mochila

- (H1)
 - * **Iniciar** com uma solução (aleatória ou com um método de construção).
 - * retirar da mochila o item que possui o menor custo.
 - * colocar na mochila o item que possui o maior custo, tal que
 - melhore o valor da função objetivo.
 - não torne a solução infactível.
 - * **Termine** quando não houver mais **movimentos** ou um determinado número fixo de movimentos tenha sido realizado

Slide 100

Busca Local - Prob. da Mochila

- (H2)
 - * **Iniciar** com uma solução (aleatória ou com um método de construção).
 - * colocar na mochila o item com maior custo que esteja fora da mochila.
 - * Se infactível, então retire os elementos com menor custo até que a solução fique factível.
 - * **Termine** quando não houver mais **movimentos** ou um determinado número fixo de movimentos tenha sido realizado

Slide 101

Busca local

- Dificuldades:
 - * Término prematuro no primeiro ótimo local encontrado
 - * Sensível à solução de partida
 - * Sensível à vizinhança escolhida
 - * Sensível à estratégia de busca
 - * Pode exigir um número exponencial de iterações!

Slide 102

Extensões para contornar algumas dificuldades da busca local

Redução da vizinhança: investigar um subconjunto da vizinhança da solução corrente (e.g. por aleatorização)

Multi-partida: repetir a busca local a partir de diferentes soluções

Multi-vizinhança: considera mais de uma vizinhança. Ao atingir um ótimo local com relação a uma vizinhança, inicia uma outra busca local empregando outra vizinhança. O algoritmo termina quando a solução corrente é um ótimo local em relação a todas as vizinhanças empregadas.

Segmentação da vizinhança: utilizada para aumentar a eficiência quando vizinhanças muito grandes são utilizadas, pode ser vista como uma estratégia multi-vizinhança. $N(s) = N_1(s) \cup N_2(s) \cup \dots \cup N_p(s)$

Slide 103

Meta-heurística

- Meta-heurísticas são modelos gerais que servem como guia para construção de algoritmos heurísticos.
- Muitos dos modelos são baseados na natureza (físicos, biológicos, evolutivos)
- Uma meta-heurística representa uma classe de heurísticas.
- As estratégias meta-heurísticas tem como objetivo superar as falhas da busca local, como por exemplo, o término prematuro em um ótimo local.

Slide 104

Algumas Meta-heurísticas

- Busca Tabu
- *Simulated Annealing*
- Algoritmos Genéticos
- *Ant System*
- Variable Neighborhood Search (VNS)
- Times Assíncronos
- GRASP (Greedy Randomized Adaptive Search Procedures)
- *Scatter Search* <http://www.densis.fee.unicamp.br/~franca/tematico/tarefas/tarefa5.html>

Slide 105

Tarefas

- Ler o primeiro capítulo de Reeves (1993)
- Faça uma pesquisa na internet com algum buscador
 - heuristics, metaheuristics
 - simulated annealing
 - tabu search
 - genetic algorithms
 - memetic algorithms
 - GRASP
- Implementar uma heurística de busca local para o PCV

Slide 106

Comentários

- Representação da Solução?
- Vizinhança? Movimento?
- Funcionamento da Heurística de Busca Local?
 - * Solução Inicial?
 - * Como será realizada a busca?
 - * Critério de parada?
- Respostas: Pensar e Realizar Testes Computacionais!!!!

Slide 107

Meta-heurística: **SIMULATED ANNEALING**

Bibliografia Básica

C. R. Reeves, Modern Heuristic Techniques for Combinatorial Problems, Blackwell, 1993.

A. Diaz, F. Glover, H. M. Ghaziri, J. L. González, M. Laguna, P. Moscato e F. T. Tseng, Optimización Heurística Y Redes Neurolales, Editorial Paraninfo, Espanha, 1996.

Simulated Annealing

- É uma analogia entre um processo de mecânica estatística e a solução de um problema de otimização combinatória
- O termo *annealing*:
 - * refere-se a um processo térmico que começa pela liquidificação de um cristal a uma alta temperatura
 - * seguido pela lenta e gradativa diminuição de sua temperatura
 - * até que o ponto de solidificação seja atingido
 - * quando o sistema atinge um estado de energia mínima

Slide 109

Simulated Annealing

- * A maneira pela qual a temperatura irá decrescer é muito importante.
- * Em um cristal muito grande, por exemplo, se a temperatura for reduzida muito rápida, o cristal conterá inúmeras imperfeições,
- * ou seja, não atinge uma configuração de energia mínima.

Slide 110

Simulated Annealing

- A heurística simulated annealing surgiu de um algoritmo denominado Metrópolis (1953), bem conhecido pelos pesquisadores da área de Física-Química.
- Kirkpatrick (1983) sugeriu que a simulação desse processo poderia ser usada para buscar soluções factíveis, com o objetivo de encontrar a solução ótima.

Slide 111

Processo físico x problema de otimização combinatória

uma solução viável \Leftrightarrow uma configuração
função objetivo $f(s)$ \Leftrightarrow nível de energia
solução vizinha \Leftrightarrow mudança de estado
parâmetro de controle \Leftrightarrow temperatura
melhor solução \Leftrightarrow estado de solidificação
solução ótima \Leftrightarrow configuração de energia mínima

Slide 112

Simulated Annealing

- Na heurística simulated annealing, são permitidos movimentos que aumentem o valor da função objetivo,
- Mas sua frequência é governada por uma distribuição de probabilidade que se altera no decorrer da heurística.
- Essa função, primeiramente, inspirou-se do processo físico, onde a probabilidade (Boltzmann) de uma certa configuração ter sua energia aumentada de um ΔE é de

$$p(\Delta E) = e^{-\Delta E / T}$$

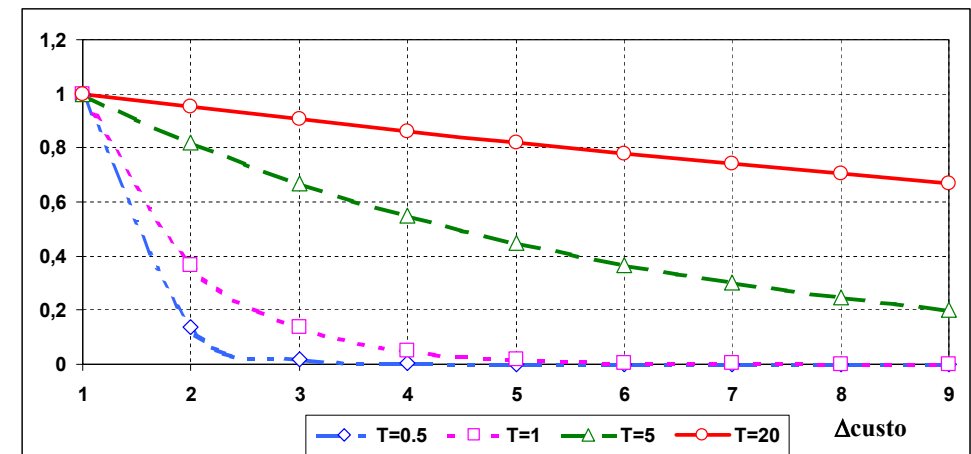
Slide 113

Simulated Annealing

- A estratégia que é utilizada no simulated annealing é:
 - * a partir de uma alta temperatura, ser permitido alterações ruins, pois estamos longe do ótimo local.
 - * posteriormente, a temperatura irá diminuindo e a possibilidade de alterações ruins vai se reduzindo, pois estamos próximos do ótimo global.

Slide 114

Probabilidade X Temperatura



Slide 115

Heurística Simulated Annealing

Entrada: T_0 , T_f , L , α (entre 0 e 1)

$T \leftarrow T_0$; $S_0 \leftarrow$ gera solução inicial; $S \leftarrow S_0$; $S^* \leftarrow S_0$

enquanto $T > T_f$ faça (temperatura alta)

para $cont \leftarrow 1$ até $L(T)$ faça (iterações para equilíbrio)

$S' \leftarrow$ seleciona uma solução vizinha de S

$\Delta custo \leftarrow custo(S') - custo(S)$

se $\Delta custo < 0$ ou $U[0,1] < \exp(-\Delta custo/T)$

então $S \leftarrow S'$

se $(S < S^*)$ então $S^* \leftarrow S$ (incumbente)

fim do para

$T \leftarrow \alpha T$

fim-enquanto

Slide 116

Parâmetros Simulated Annealing

- Parâmetros do simulated annealing

- * T_0 = temperatura inicial.

- * T_f = temperatura final.

- * L = número de iterações para atingir o equilíbrio em uma dada temperatura.

- * α = proporção de redução da temperatura.

- Um dos parâmetros pode ser a função de aceitação de uma configuração ruim.

- Os parâmetros mais adequados para uma dada aplicação só podem ser estabelecidos por experimentação**

Slide 117

Parâmetros Simulated Annealing - Temperatura

- Temperatura Inicial

- * chute total, mas um valor suficientemente alto para que soluções ruins sejam aceitas com bastante frequência no início da heurística.

- * chute, sendo que o seu valor faça com que por volta de 50% de soluções ruins sejam aceitas.

- * relacionado ao valor da função objetivo.

- (Diaz et al., 1996), página 44.

- baseado na variação do valor da função objetivo na primeira fase da heurística.

Slide 118

Parâmetros Simulated Annealing - Temperatura

- Processo de redução (alteração) da temperatura

- * essa escolha depende de quanto se quer explorar determinadas regiões.

- * através do parâmetro α (0.8 a 0.99).

- * é atualizada com o valor da variação do custo na última etapa de busca.

- * atualização

$$T_0 = \text{Custo}(S_1) - \text{Custo}(S_0)$$

$$T_{i+1} = [T_i + \text{Custo}(S_{i+1}) - \text{Custo}(S_i)]/2$$

(ou ainda a média feita durante o período de equilíbrio)

Slide 119

Parâmetros Simulated Annealing - Temperatura

- Número de iterações para atingir o equilíbrio em uma dada temperatura.
 - * número fixo de iterações.
 - * depois de um certo tempo/iteraões sem alterar o valor da função objetivo.
 - * o número de iterações/tempo pode estar relacionado com o tamanho do problema.
 - * pode também estar relacionado com a vizinhança escolhida.

Slide 120

Parâmetros Simulated Annealing - Temperatura

- temperatura final
 - * ≈ 0
 - * limitação por tempo ou iterações.
 - * finalização depois de um certo tempo/iteraões sem que haja atualização da solução corrente.
 - * esse tempo/iteraões fixo pode estar relacionado com o tamanho do problema .
 - * pode também estar relacionado com a vizinhança escolhida.

Slide 121

Parâmetros Simulated Annealing - Função para aceitação de uma configuração ruim

- Função básica
 $\exp(-\Delta \text{custo}/T)$
- Funções determinísticas



- (1) aceita soluções ou “muito boas” ou “pouco ruins”
- (2) rejeita soluções “muito ruins”.

Slide 122

Vantagens

- Existe prova de convergência para a solução ótima
- Implementação simples
 - * só visita uma solução a cada iteração
 - * bastando calcular o valor da função objetivo da solução vizinha gerada

Slide 123

Desvantagens

- Apesar de convergir para a solução ótima, a velocidade de redução de temperatura exigida implica em visitar um número exponencial de soluções.
- Em princípio é necessário um processo lento de redução da temperatura e isso resulta em tempos de processamento elevados.
- Pouco “inteligente”, pois utiliza como informação do problema somente a variação do valor da função objetivo.
- Muitos parâmetros para calibrar.

Slide 124

Comentários

- Pela simplicidade de implementação, pode ser utilizado em conjunto com alguma outra heurística ou outra meta-heurística.
- Existem implementações, onde apenas a idéia de *simulated annealing* é utilizado para melhorar o desempenho de outra heurística/metaheurística, como por exemplo, embutir a estratégia de aceitar soluções ruins com certa probabilidade.

Slide 125

Comentários

Dois importantes aspectos para o sucesso de uma implementação de simulated annealing:

- vizinhança adotada
- estratégia de resfriamento

Slide 126

Exemplo - sequenciamento

- Seqüenciar 4 tarefas em uma máquina com o objetivo de minimizar o atraso
 - * Tempos de processamento = (6 4 8 2)
 - * Datas de entrega = (9 12 15 8)

Slide 127

Exemplo - sequenciamento

- Sequenciar 4 tarefas em uma máquina com o objetivo de minimizar o atraso
 - * Tempos de processamento = (6 4 8 2)
 - * Datas de entrega = (9 12 15 8)
- Suponha a solução $S = (3\ 2\ 1\ 4)$, construída aleatoriamente (poderíamos pensar em uma heurística de construção mais elaborada)
 - * término = (8 12 18 20)
 - * atraso = $0 + 0 + 9 + 12 = 21$

Slide 128

Vizinhança

Solução $\pi = (\pi_1, \dots, \pi_{i-1}, \pi_i, \pi_{i+1}, \dots, \pi_j, \dots, \pi_n)$

$N1(\pi) = \{(\pi_1, \dots, \pi_{i+1}, \pi_i, \dots, \pi_n) : i=1, \dots, n-1\}$

Vizinhos de $(1, 2, 3, 4) = \{(2, 1, 3, 4), (1, 3, 2, 4), (1, 2, 4, 3)\}$

- Ou seja, trocar duas tarefas consecutivas

Slide 129

Exemplo - sequenciamento

- $S = (3\ 2\ 1\ 4)$, atraso = 21

i	j	valor movimento	atraso
3	2	0	21
2	1	+2	23
1	4	-4	17



Slide 130

Exemplo - sequenciamento

- Qual movimento realizar?
 - * Se a cada iteração forem analisados todos os movimentos possíveis, então será o $i=1$ e $j=4$.
 - * Se for realizado o primeiro movimento que melhora (não piora) a função objetivo então será $i=3$ e $j=2$.
- Vamos supor que todos os movimentos estão sendo analisados. Então
 - * $S = (3\ 2\ 4\ 1)$, atraso = 17

Slide 131

Exemplo - sequenciamento

- $S = (3\ 2\ 4\ 1)$, atraso = 17

i	j	valor movimento	atraso
3	2	0	17
2	4	-2	15
4	1	+4	21

- $S = (3\ 4\ 2\ 1)$, atraso = 15

Slide 132

Exemplo - sequenciamento

- $S = (3\ 4\ 2\ 1)$, atraso = 15

i	j	valor movimento	atraso
3	4	-6	9
4	2	+2	17
2	1	+2	17

- $S = (4\ 3\ 2\ 1)$, atraso = 9

Slide 133

Exemplo - sequenciamento

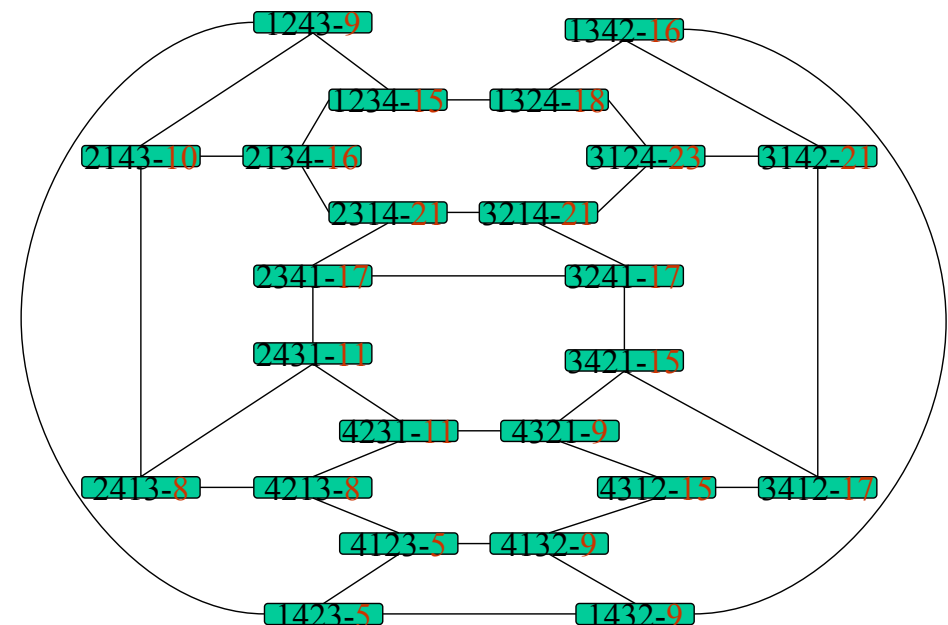
- $S = (4\ 3\ 2\ 1)$, atraso = 9

i	j	valor movimento	atraso
4	3	+6	15
3	2	+2	11
2	1	+6	15

- Ótimo Local!!!

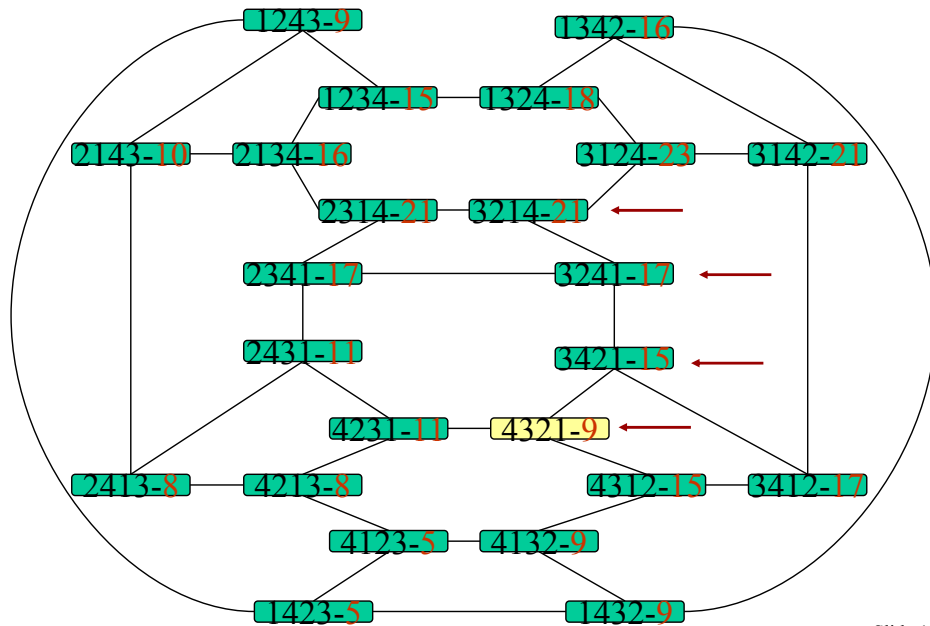
Slide 134

Vizinhança/Espaço de Busca



Slide 135

Vizinhança/Espaço de Busca



Slide 136

Simulated Annealing

- Poderíamos aplicar nessa heurística de busca local, simulated annealing (SA).
- Com SA sairíamos do ótimo local, pois movimentos que permitem soluções ruins são aceitos, mas com certa probabilidade.

Slide 137

Exemplo - sequenciamento

- $S = (4\ 3\ 2\ 1)$, atraso = 9

i	j	valor movimento	atraso
4	3	+6	15
3	2	+2	11
2	1	+6	15



- Aceitar a Solução $S = (4\ 2\ 3\ 1)$, atraso = 11

Slide 138

Exemplo - sequenciamento

- $S = (4\ 2\ 3\ 1)$, atraso = 11

i	j	valor movimento	atraso
4	2	0	11
2	3	-2	9
3	1	-3	8



- $S = (4\ 2\ 1\ 3)$, atraso = 8

Slide 139

Exemplo - sequenciamento

- $S = (4\ 2\ 1\ 3)$, atraso = 8

i	j	valor movimento	atraso
4	2	0	8
2	1	-3	5
3	1	+3	11

- $S = (4\ 1\ 2\ 3)$, atraso = 5

Slide 140

Exemplo - sequenciamento

- $S = (4\ 1\ 2\ 3)$, atraso = 5

i	j	valor movimento	atraso
4	1	0	5
1	2	+3	8
2	3	+4	9

- $S = (1\ 4\ 2\ 3)$, atraso = 5

Slide 141

Exemplo - sequenciamento

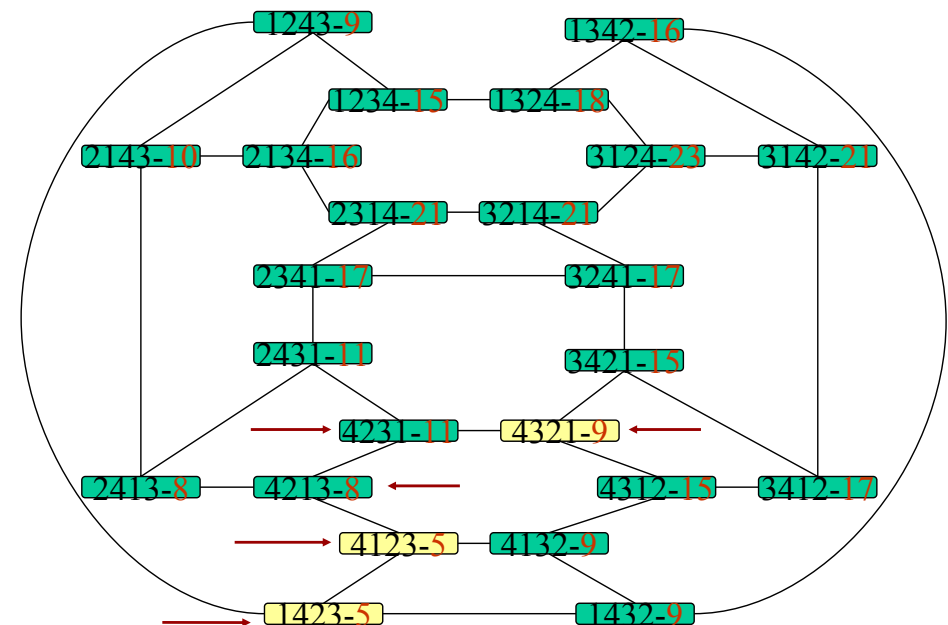
- $S = (1\ 4\ 2\ 3)$, atraso = 5

i	j	valor movimento	atraso
1	4	0	5
4	2	+4	9
2	3	+4	9

- O Algoritmo entra em ciclagem!!
- Logo: $S = (1\ 4\ 2\ 3)$ ou $S = (4\ 1\ 2\ 3)$ seria as soluções encontradas. Neste caso, soluções ótimas

Slide 142

Vizinhança/Espaço de Busca



Slide 143

Exercício

- Acrescentar ao programa um procedimento de melhoria da solução utilizando Simulated Annealing
- Implementar mecanismos de comparação entre as soluções obtidas por diferentes procedimentos heurísticos

Slide 144

Comentários

- A implementação deve utilizar o procedimento de busca local acrescido das características da SA... isso permite uma comparação mais justa

Slide 145

Busca Tabu

Silvio Alexandre de Araujo

Slide 146

BUSCA TABU

- M. Laguna, Tabu Search Tutorial, II Escuela de Verano Latino-Americana de Investigación Operativa, 1995.
- A. Diaz, F. Glover, H. M. Ghaziri, J. L. González, M. Laguna, P. Moscato e F. T. Tseng, Optimización Heurística Y Redes Neurolales, Editorial Paraninfo, Espanha, 1996.

Slide 147

Princípios gerais

- Proposto por Fred Glover, 1986.
- Guia para busca local
- Seu principal componente é o uso de memória adaptativa para criar uma busca mais flexível.
 - * analogia → mountain climbing → precisa lembrar (memória) elementos chave para ser capaz de tomar decisões ao longo do caminho.
 - * busca inteligente

Slide 148

Princípios gerais

- contrasta com métodos que:
 - * não utilizam memória (*simulated annealing*)
 - * utilizam estruturas rígidas de memória (*branch-and-bound*).
 - * memória de atributos de configurações visitadas no passado é usado para proibir movimentos que levariam a configurações já visitadas.
 - * restrição de atributos de soluções -> restrições de movimentos -> necessidade de estrutura de dados eficiente de memória.

Slide 149

Uso da memória

- As estruturas de memória tem quatro princípios:
 - * recentidade, frequência → utilizada em estratégias mais “sofisticadas” de tabu.
 - * qualidade → habilidade de diferenciar o mérito de uma solução durante a busca. Podendo identificar elementos que são comuns entre as melhores soluções e elementos que fazem parte das ruins.
 - * influência → considera o impacto dessas escolhas durante a busca.

Slide 150

Uso da memória

- O uso de memória pode ser explícito ou através de atributos.
 - * explícito armazena soluções completas, por exemplo, armazenando soluções de elite visitadas durante a busca
 - * atributos de soluções que se modificam depois de alguns movimentos
 - grafo → adição e/ou retirada de uma aresta.
 - scheduling → índice das tarefas.

Slide 151

Uso da memória

- As estratégias com o uso da memória estão divididas em

- * memória de curto prazo
 - recentidade
- * memória de longo prazo
 - intensificação e diversificação
 - frequência

Slide 152

Busca Local

inicie com uma solução S

faça

$melhora \leftarrow 0$

$S' \leftarrow$ seleciona

$\Delta custo \leftarrow custo(S') - custo(S)$

se $\Delta custo < 0$ então

$S \leftarrow S'$

$melhora \leftarrow 1$

enquanto ($melhora$)

Slide 153

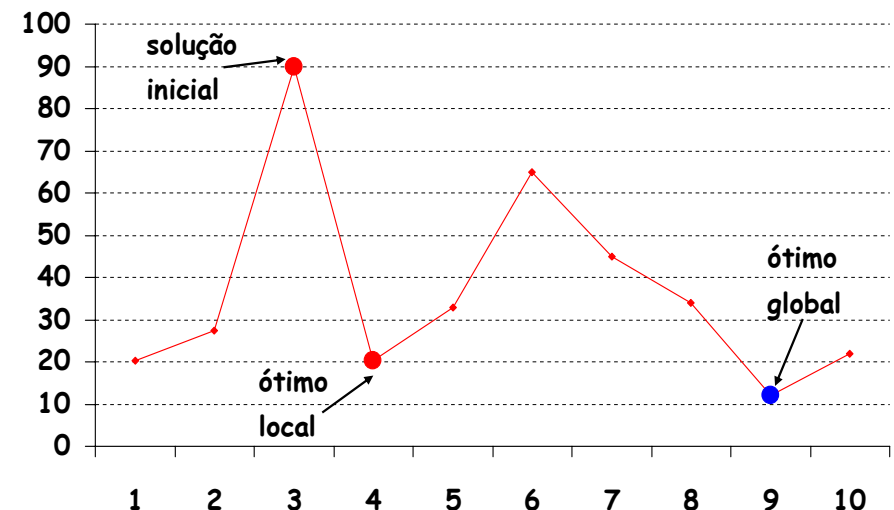
Busca Local

- seleciona pode ser:

- * (H1) - Analise todas as soluções vizinhas e escolha a que leve a menor $\Delta custo$
 - analisar todas pode ser computacional/e muito caro
- * (H2) - Analise todas as soluções vizinhas e escolha a que leve a menor $\Delta custo$, mas com $\Delta custo < 0$.
- * (H3) - Analise as soluções vizinhas e escolha a primeira que leve a $\Delta custo < 0$.

Slide 154

Busca Local



Slide 155

Busca Local - exemplo - sequenciamento

- Sequenciar 4 tarefas em uma máquina com o objetivo de minimizar o atraso
 - * Tempos de processamento = (6 4 8 2)
 - * Datas de entrega = (9 12 15 8)
- Suponha a solução $S = (1\ 2\ 3\ 4)$, construída aleatoriamente (poderíamos pensar em uma heurística de construção mais elaborada)
 - * término = (6 10 18 20)
 - * atraso = $0 + 0 + 3 + 12 = 15$

Slide 156

Busca Local - exemplo - sequenciamento

- $S = (1\ 2\ 3\ 4)$, atraso = 15
- vizinhança: trocar a tarefa i com a tarefa j

i	j	valor movimento	atraso	
1	2	+1	16	
1	3	+6	21	
1	4	-4	11	
2	3	+3	18	
2	4	-6	9	←
3	4	-6	9	

Slide 157

Busca Local - exemplo - sequenciamento

- Qual movimento realizar?
 - * Se a cada iteração forem analisados todos os movimentos possíveis, então será o $i=2$ e $j=4$.
 - * Se for realizado o primeiro movimento que melhora a função objetivo então será $i=1$ e $j=4$.
- Vamos supor que todos os movimentos estão sendo analisados. Então
 - * $S = (1\ 4\ 3\ 2)$, atraso = 9

Slide 158

Busca Local - exemplo - sequenciamento

- $S = (1\ 4\ 3\ 2)$, atraso = 9

i	j	valor movimento	atraso	
1	2	+2	11	
1	3	+8	17	
1	4	0	9	
2	3	-4	5	←
2	4	+6	15	
3	4	+7	16	


- $S = (1\ 4\ 2\ 3)$, atraso = 5

Slide 159

Busca Local - exemplo - sequenciamento

- $S = (1\ 4\ 2\ 3)$, atraso = 5

i	j	valor movimento	atraso
1	2	+3	8
1	3	+10	15
1	4	0	5
2	3	+4	9
2	4	+4	9
3	4	+13	18




- $S = (4\ 1\ 2\ 3)$, atraso = 5
- Pensando na heurística de busca H1.

Slide 160

Busca Local - exemplo - sequenciamento

- $S = (4\ 1\ 2\ 3)$, atraso = 5

i	j	valor movimento	atraso
1	2	+3	8
1	3	+8	13
1	4	0	5
2	3	+4	9
2	4	+5	10
3	4	+18	23



- Entrou em ciclagem

Slide 161

Simulated Annealing

- Poderíamos aplicar nessa heurística de busca local, simulated annealing (SA).
- Com SA sairíamos da ciclagem, pois movimentos que permitem soluções ruins são aceitas, mas com certa probabilidade.
- Poderíamos aplicar em qualquer das heurísticas de busca (H1, H2 e H3).

Slide 162

Simulated Annealing x Busca Tabu

- O SA e algumas outras estratégias metaheurísticas aplicam randomização para fugir de um ótimo local e da ciclagem.
- Os “usuários” de Busca Tabu argumentam que tais estratégias são pouco inteligentes.

Slide 163

Busca Tabu - memória de curto prazo

- A estratégia é de **classificar como proibido** certos movimentos ou soluções, através de atributos, armazenando-os na **lista tabu** durante um certo **tempo**.
- Para tornar a busca mais flexível também temos como desclassificar um movimento de tabu como não tabu por algum **critério**.

Slide 164

Busca Tabu - memória de curto prazo

- Elementos
 - * O que proibir e como:
 - Lista Tabu
 - Atributo
 - Regra de ativação tabu
 - * Por quanto tempo
 - Tempo Tabu
 - * Critério de Aspiração

Slide 165

Memória de curto prazo - lista tabu e atributo

- Inicialmente é preciso definir como será caracterizado o que é tabu, ou seja, a escolha de um atributo e como é feita a regra de ativação tabu.
- Essa decisão envolve
 - * escolha de estrutura de dados.
 - * o tipo de vizinhança adotada na heurística de busca.
 - * a representação da solução.
 - * como será feita a verificação se certo movimento é tabu

Slide 166

Memória de curto prazo - lista tabu e atributo

- Nosso exemplo seqüenciamento
 - * movimento = sai da solução S1, troca i com j, temos a solução S2.
 - * depois de feito esse movimento poderíamos ter como proibido o que?
 - ir a solução S1.
 - trocar i com j, ou vice-versa.
 - trocar i com alguém (ou j com alguém).

Slide 167

Memória de curto prazo - lista tabu e atributo

- ir a solução S1.
 - * atributo → uma solução inteira, um vetor.
 - * lista tabu → lista de soluções
- trocar i com j, ou vice-versa.
 - * atributo → (i,j)
 - * lista tabu → matriz nxn
- trocar i com alguém (ou j com alguém).
Atributo
 - * atributo → i
 - * lista tabu → vetor n

Slide 168

Memória de curto prazo - lista tabu e atributo

- Planejamento da produção de N itens em T períodos
 - * solução → x_{it} = quantidade
 - * vizinhança : mover uma quantidade q da produção do item i do período t para o período t'.
- atributo
 - * (q,i,t,t') → proibir q de i de sair de t' e ir para t
 - * (i,t,t') → proibir i de sair de t' e para ir a t.
 - * (i,t) → proibir i de sair (ou ir para) t.

Slide 169

Memória de curto prazo - lista tabu e atributo

- Problema da mochila
 - * solução → $x_i = 0$ ou 1
 - * vizinhança
 - tirar o item i da mochila ($x_i=1 \rightarrow x_i=0$)
 - colocar o item i na mochila ($x_i=0 \rightarrow x_i=1$)
- atributo
 - * solução (vetor) → ir a uma solução S.
 - * (i) → proibir i de sair (ou entrar) na mochila.

Slide 170

Memória de curto prazo - tempo tabu

- Tempo tabu
 - * número fixo
 - descoberto experimentalmente.
 - relacionado com o tamanho do problema
 - relacionado com a vizinhança adotada
 - todas as anteriores
 - * número sorteado aleatoriamente entre [Tmin,Tmax]
 - Tmin e Tmax escolhidos como o número fixo
 - * o número fixo ou o Tmin e Tmax podem variar durante a busca

Slide 171

Memória de curto prazo - critério de aspiração

- Esse critério é utilizado para liberar um movimento de seu status tabu antes do seu tempo tabu terminar.
- Por exemplo,
 - * Se um certo movimento tabu levar a uma solução melhor que a incumbente (melhor encontrada até o momento) libere-o.
 - * Se todos os movimentos possíveis de serem realizados estão tabu, libere um deles segundo algum critério.

Slide 172

Exemplo - sequenciamento

- Solução inicial: $S = (1\ 4\ 3\ 2)$, atraso = 9
- Busca local
 - * movimento: trocar i com j
 - * analise todas as soluções vizinhas e escolha a que leve a menor Δ custo
- Busca Tabu
 - * atributo (i,j) , ou seja, proibido de trocar i com j . (estrutura de dados)
 - * tempo tabu: ?
 - * critério: quando for atingida uma solução melhor que a incumbente.

Slide 173

Exemplo - sequenciamento

- $S = (1\ 4\ 3\ 2)$, atraso = 9

i	j	valor movimento	atraso
1	2	+2	11
1	3	+8	17
1	4	0	9
2	3	-4	5
2	4	+6	15
3	4	+7	16

- $S = (1\ 4\ 2\ 3)$, atraso = 5
- $TABU = \{(2,3)\}$, $atraso^* = 5$

Slide 174

Exemplo - sequenciamento

- $S = (1\ 4\ 2\ 3)$, atraso = 5

i	j	valor movimento	atraso
1	2	+3	8
1	3	+10	15
1	4	0	5
2	3	+4	9
2	4	+4	9
3	4	+13	18

- $S = (4\ 1\ 2\ 3)$, atraso = 5
- $TABU = \{(2,3), (1,4)\}$, $atraso^* = 5$

Slide 175

Exemplo - sequenciamento

- $S = (4\ 1\ 2\ 3)$, atraso = 5

i	j	valor movimento	atraso
1	2	+3	8 ←
1	3	+8	13
1	4	0	5 ← Tabu
2	3	+4	9
2	4	+5	10
3	4	+18	23

- $S = (4\ 2\ 1\ 3)$, atraso = 8
- TABU = {(2,3), (1,4), (2,1)}, atraso* = 5

Slide 176

Exemplo - sequenciamento

- $S = (4\ 2\ 1\ 3)$, atraso = 8

i	j	valor movimento	atraso
1	2	-3	5 ← Tabu
1	3	+3	11
1	4	+1	9
2	3	+7	15
2	4	0	8 ←
3	4	+13	21

- $S = (2\ 4\ 1\ 3)$, atraso = 8
- TABU = {(2,3), (1,4), (2,1), (2,4)}, atraso* = 5

Slide 177

Exemplo - sequenciamento

- E assim continua a busca...
- É preciso atualizar a lista, dependendo do tipo de estrutura de dados utilizada. Nesse caso poderíamos usar uma matriz.
- Cada movimento que descartávamos, pois era tabu, poderia ter sido feito se levasse a uma solução com atraso menor que 5 (critério de aspiração)

Slide 178

Outros elementos - Lista de Candidatos

- Lista de candidatos → restringir o número de elementos a serem analisados
 - * vizinhança número de candidatos muito grande.
 - analisar apenas as trocas de tarefas onde pelo menos uma delas tem atraso.
 - selecionar aleatoriamente um vizinho.
 - selecionar o primeiro movimento que passe por certo critério.
 - * avaliação “cara” do candidato.
 - influência alta da estrutura de dados
- Outros elementos existem...

Slide 179

Memória de longo prazo

- modificação da vizinhança para incluir soluções que nunca apareceram
 - * selecionar soluções de elite (ótimos locais de alta qualidade)
 - * tentar conseguir “ver a cara” das soluções visitadas até agora.
 - * tentar conseguir “ver a cara” das soluções de boa qualidade (soluções de elite) visitadas até agora.

Slide 180

Memória de longo prazo - medida de frequência

- Contabilizar informações durante a busca tabu de curto prazo.
- Razões
- Contadores de dois tipos de ocorrência:
 - * **residência** → número de vezes que o atributo esteve em um conjunto de soluções
 - análise de um conjunto de soluções
 - * **transição** → número de vezes que um atributo trocou.
 - análise dos passos da heurística

Slide 181

Memória de longo prazo - medida de frequência

- denominador → quantidade
 - * total número de soluções
 - * soma dos numeradores
 - * maior número no numerador
 - * a média dos numeradores

Slide 182

Exemplo - scheduling

- Suponha os seguintes movimentos:

(4,2)	1 4 3 2
(2,3)	1 4 2 3
(4,1)	4 1 2 3
(2,1)	4 2 1 3
(2,4)	2 4 1 3
(3,1)	2 4 3 1
(3,2)	3 4 2 1
(4,3)	4 3 2 1
(1,2)	4 3 1 2
(1,3)	4 1 3 2

residência

2	2	3	3
2	1	4	3
1	2	3	4
5	5	0	0

transição

5	6	4	4
---	---	---	---

apenas os numeradores

Slide 183

Exemplo - scheduling

- Supondo o denominador como sendo o número de soluções = 10

residência

0.2 0.2 0.3 0.3

0.2 0.1 0.4 0.3

0.1 0.2 0.3 0.4

0.5 0.5 0.0 0.0

transição

0.5 0.6 0.4 0.4

matriz residência
de todas as soluções

Slide 184

Memória de longo prazo - medida de frequência

- construímos a matriz e o vetor utilizando todas as soluções durante a busca.
- Poderíamos também utilizar um subgrupo de soluções
 - * soluções de elite
 - * em heurísticas composta por diferentes procedimentos, poderíamos construir tais matrizes utilizando apenas soluções produzidas por específicos procedimentos.

Slide 185

Estratégia de intensificação

- a idéia é intensificar a busca em regiões onde parece ter maiores chances de apresentar soluções de qualidade.
 - * modificar regras de escolha de forma a encorajar
 - movimentos e soluções com características “boas” no decorrer da história da busca.
 - combinar movimentos com a estrutura dessas soluções
 - * reiniciar a busca de uma solução “boa”.

Slide 186

Estratégia de diversificação

- a idéia é tentar explorar regiões ainda inexploradas ou pouco exploradas
 - * modificando as regras de escolha de forma a
 - desencorajar movimentos e soluções muito frequentes no decorrer da história da busca (penalizações)
 - incentivar movimentos e soluções pouco frequentes no decorrer da história da busca (penalizações)
 - * reiniciar a busca utilizando as informações de frequência

Slide 187

medida frequência x intensificação e diversificação

- atributos com
 - * alta frequência → a partir das soluções de elite → atributos atrativos → **intensificação**
 - * baixa frequência → a partir das soluções de elite → atributos atrativos → **diversificação**
 - * baixa frequência → feita a partir de todas as soluções → atributos atrativos → **diversificação**
 - * etc.

Slide 188

estratégias de intensificação

- Reiniciar a busca a partir de uma “boa” solução.
- Reiniciar a busca a partir de uma solução que possua o maior número de características de medida de residência.
- Reiniciar a busca a partir de uma solução construída levando-se em conta as características de medida de residência.

Slide 189

Matriz residência para as soluções de elite

- Suponha os seguintes movimentos:

(4,2)	1 4 3 2	9
(2,3)	1 4 2 3	5
(4,1)	4 1 2 3	5
(2,1)	4 2 1 3	8
(2,4)	2 4 1 3	8
(3,1)	2 4 3 1	11
(3,2)	3 4 2 1	15
(4,3)	4 3 2 1	13
(1,2)	4 3 1 2	15
(1,3)	4 1 3 2	9

1 1 2 0
1 1 2 0
0 0 0 3
2 2 0 0

residência
0.3 0.3 0.5 0.0
0.3 0.3 0.5 0.0
0.0 0.0 0.0 0.8
0.5 0.5 0.0 0.0

Slide 190

estratégias de diversificação

- Para a escolha do movimento
 $\text{valor_movimento}' = \text{valor_movimento} + d * \text{penalidade}$
- Para reiniciar a busca
 - * construção de uma nova solução de partida.
 Na sua heurística de construção mude a prioridade da tarefa utilizando uma penalidade proporcional à medida de frequência
 $\text{prioridade_tarefa}' = \text{prioridade_tarefa} + d * \text{penalidade}$
- O valor da penalidade utilizada nas duas expressões acima tem que dar prioridade às tarefas pouco frequentes.

Slide 191

Oscilação estratégica

- alternar a busca entre região factível e região infactível
 - * deixando a solução ser infactível até certo “ponto” e a seguir, através de penalidades, voltar à região factível.
- alternar entre diversificação e intensificação

Slide 192

Path relinking

- intensificação da busca no caminho que leva a soluções de elite
 - * armazenar um conjunto de soluções de elite
 - * para cada par de soluções de elite:
 - identificar as diferenças entre elas
 - executar um procedimento de busca local sem parar em nenhum ótimo local, executando sempre o melhor movimento dentre aqueles ainda não executados
 - verificar a existência nesta trajetória de soluções melhores do que as extremidades

Slide 193

Tarefa

- Implementar Busca Tabu para o problema do caixeiro viajante com pelo menos as seguintes estratégias:
 - * curto prazo
 - atributo (teste pelo menos 2 tipos)
tamanho da lista (testes)
critério de aspiração.
 - * longo prazo
 - pelo menos uma medida de frequência (por exemplo, residência das soluções de elite) para implementação de pelo menos uma estratégia de intensificação e uma de diversificação.

Slide 194

Tarefa

- A cada novo elemento implementado façam testes para analisar o seu efeito.
- Pensem que vocês vão implementar pelo menos 4 tipos de meta-heurísticas onde cada uma delas terá várias versões
- Vocês precisarão fazer uma prova e entregar um “artigo” com as implementações, testes, análises e conclusões

Slide 195

Algoritmos Genéticos

Silvio Alexandre de Araujo

Slide 196

ALGORITMOS GENÉTICOS

- A. Diaz, F. Glover, H. M. Ghaziri, J. L. González, M. Laguna, P. Moscato e F. T. Tseng, Optimización Heurística Y Redes Neurolales, Editorial Paraninfo, Espanha, 1996.
- C. R. Reeves, Modern Heuristic Techniques for Combinatorial Problems, Blackwell, 1993.
- Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-verlag, 1992.
- M. Gen e R. Cheng, Genetic Algorithms & Engineering Design, Wiley, 1997.

Slide 197

Evolução

- **população** → grupo de organismos vivos
- maior capacidade de adaptação → maior chance de sobreviver
- os sobreviventes vão se reproduzir → gerando uma nova população

Slide 198

Evolução

- características hereditárias são herdadas → **cruzamento entre os cromossomos**
- cromossomos contêm informações denominados genes, que por sua vez são compostos por valores chamados de alelos.
- genes podem se alterar de uma geração para outra → **mutação**

Slide 199

Evolução x Problemas de otimização

- Introduzido por Holland, 1973
- princípio → sobrevivência do mais capaz
- Inicialmente, algoritmos simples, mas conseguiam boas soluções para problemas considerados difíceis naquela época.

Slide 200

Glossário

- **indivíduo** (genotype, cromossomo, estrutura, string, etc.)
 - * cadeia de genes que codifica uma possível configuração do problema.
 - * codificação de uma solução (existem várias maneiras de codificar uma mesma solução).

Slide 201

Glossário

- **gene** (caracter)
 - * representa uma certa característica da solução-cromossomo
- **allele**
 - * um particular estado de um gene
- **locus**
 - * posição de um gene em um cromossomo

Slide 202

Glossário

- **população**
 - * um conjunto de indivíduos (cromossomos) representando os atuais pontos que fazem parte da busca no espaço de soluções.
- **geração**
 - * uma maneira de identificar diferentes populações no processo de evolução.

Slide 203

Glossário

- função de **fitness** (capacidade)
 - * função utilizada pelo esquema de seleção
 - * geralmente está ligado ao valor da função objetivo
 - * pode incluir outros elementos (penalização pela violação de restrições)

Slide 204

Glossário

- operador de reprodução (**crossover**)
 - * uma função/ algoritmo que recebe como input um subconjunto da população e cria outro conjunto
 - * geralmente 2 indivíduos para criação de 1 ou 2 filhos.
- **mutação**
 - * pequena perturbação aleatória que modifica um cromossomo.

Slide 205

Características

- os cromossomos que codificam indivíduos mais “capazes”, se reproduzem com maior probabilidade
- a mutação produz filhos com cromossomos diferentes dos pais.

Slide 206

Características

- a evolução não tem memória
- utiliza um grupo de soluções
- não partem de uma solução e uma aplicação posterior de movimentos (simulated annealing e busca tabu)

Slide 207

Aspectos importantes para um bom GA

- boa representação
- população inicial (aleatória ou construtiva?)
- função de fitness
- bom/eficiente operador genético
 - * crossover e mutação
- parâmetros

Slide 208

Um GA simples

- 1) **Inicialize** a população
- 2) Para todos os membros da população corrente avalie a função de **fitness**
- 3) Crie uma nova população de cromossomos
 - * usando **crossover** e **mutação**
- 4) Crie uma **nova população** adicionando os indivíduos criados em 3)
- 5) Se a **condição de parada** não for satisfeita, retorne a 2)

Slide 209

Representação de um indivíduo

- string de 0 e 1's (mochila, partição de números)
0 1 0 1 1 1 0
- string de números inteiros (seqüenciamento)
2 4 5 3 1 6

Slide 210

População inicial

- Aleatória
- Usando uma heurística de construção
 - * com posterior perturbação para gerar diferentes indivíduos
 - * com perturbação anterior aos dados de entrada da heurística para construção de diferentes soluções

Slide 211

População: tamanho e estrutura

- tamanho
 - * pequeno x grande
- estrutura
 - * os indivíduos podem ser independentes entre si.
 - * podem estar conectados, com uma definição mais precisa da escolha dos pais. (ex. árvore)

Slide 212

Função de fitness

- valor da função objetivo
 $f(x)$
- valor da função objetivo + penalidade pela violação de um conjunto de restrições
 $f(x) + a \cdot p(x)$
 $f(x) \cdot p(x)$
- valor da função objetivo + medida de diversidade
 $f(x)/d(x)$
 $f(x) + d(x)$

Slide 213

Reprodução: escolha dos pais

- aleatória
- aleatória relacionada com o valor de fitness (veremos no exemplo)
- relacionado com o valor de fitness
- quando há estrutura, isso já está de uma certa maneira determinado

Slide 214

Operadores de reprodução (crossover)

- one-point crossover


pai1 (0 1 0 1 1 0 1 0 0) pai2 (1 1 0 0 1 0 0 1 0)

filho1 (0 1 0 0 1 0 0 1 0) filho2 (1 1 0 1 1 0 1 0 0)

Slide 215

Operadores de reprodução (crossover)

- two-point crossover

pai1 (0 1 0 1 1 0 1 0 0) pai2 (1 1 0 0 1 0 0 1 0)

 filho1 (0 1 0 0 1 0 1 0 0) filho2 (1 1 0 1 1 0 0 1 0)

Slide 216

Operadores de reprodução (crossover)

- crossover uniforme

pai1 (0 1 0 1 1 0 1 0 0) pai2 (1 1 0 0 1 0 0 1 0)
 filho (x 1 0 x 1 0 x x 0)

pai1 (0 1 0 1 1 0 1 0 0) pai2 (1 1 0 0 1 0 0 1 0)
 filho1 (0 1 0 0 1 0 0 0 0) filho2 (1 1 0 1 1 0 1 1 0)

Slide 217

Operador de reprodução (crossover)

- Exemplos para representação não binária

(1 2 4 6 3 5) (2 3 4 1 6 5)
 (x x 4 x x 5)

(1 2 4 6 3 5) (2 3 4 1 6 5)
 (2 3 4 6 3 5)
 (2 * 4 6 3 5)
 (2 1 4 6 3 5)

Slide 218

Operador de mutação

- Troque o valor de cada bit com uma probabilidade de 10%.
- Troque dois elementos de posição com probabilidade de 10%

(1 0 1 0 1 1)
 (1 0 1 0 0 1)

(2 3 4 1 6 5)
 (1 3 4 2 6 5)

Slide 219

Nova população

- Colocar o filho no lugar do pior pai
- “Matar” os indivíduos com menor fitness.
 - * depois da criação de um novo filho
 - * depois da população atingir um número k de indivíduos.
- Com população usando estrutura, isso já está definido

Slide 220

Critério de parada

- tempo de execução
- número de gerações
- perda de diversidade
- convergência
 - * nas últimas k iterações não houve melhora da incumbente.

Slide 221

Exemplo 1

- Problema: $\max f(x) = x^2$, $x \in [0,31]$ e inteiro
- **representação** → cada inteiro como um cromossomo de 5 bits
 - $0 = (0,0,0,0,0)$
 - $31 = (1,1,1,1,1)$
- **população inicial** → aleatória
- população de tamanho 4
- função de **fitness** → $f(x)$

Slide 222

Exemplo 1

- População inicial gerada:

	x	f(x)	
$A_1 \rightarrow$	0 1 1 0 1	13	169
$A_2 \rightarrow$	1 1 0 0 0	24	576
$A_3 \rightarrow$	0 1 0 0 0	8	64
$A_4 \rightarrow$	1 0 0 1 1	19	361
			← Pai 1
			← Pai 2

- Escolha dos pais com maior fitness
- geração de 1 filho, que substituirá o indivíduo com menor fitness

Slide 223

Exemplo 1

	x	f(x)
Pai ₁ = A ₂	→ 1 1 0 0 0	24 576
Pai ₂ = A ₄	→ 1 0 0 1 1	19 361
Filho	→ 1 1 0 1 1	
mutação	→ 1 1 0 0 1	25 625

	x	f(x)	
A ₁	→ 0 1 1 0 1	13 169	
A ₂	→ 1 1 0 0 0	24 576	
A ₃	→ 1 1 0 0 1	25 625	← Pai 2
A ₄	→ 1 0 0 1 1	19 361	← Pai 1

Slide 224

Exemplo 1

	x	f(x)
Pai ₁ = A ₂	→ 1 1 0 0 0	24 576
Pai ₂ = A ₃	→ 1 1 0 0 1	25 625
Filho	→ 1 1 0 0 1	
mutação	→ 1 1 0 1 1	27 729

	x	f(x)	
A ₁	→ 1 1 0 1 1	27 729	
A ₂	→ 1 1 0 0 0	24 576	← Pai 1
A ₃	→ 1 1 0 0 1	25 625	← Pai 2
A ₄	→ 1 0 0 1 1	19 361	

Slide 225

Exemplo 1

	x	f(x)
Pai ₁ = A ₁	→ 1 1 0 1 1	27 729
Pai ₂ = A ₃	→ 1 1 0 0 0	24 576
Filho	→ 1 1 0 0 0	
mutação	→ 1 1 1 0 0	28 784

	x	f(x)
A ₁	→ 1 1 0 1 1	27 729
A ₂	→ 1 1 0 0 0	24 576
A ₃	→ 1 1 0 0 1	25 625
A ₄	→ 1 1 1 0 0	28 784

e assim continua...

Slide 226

Aspectos importantes!!!

Representação

População

Crossover

Mutação

Fitness

Slide 227