NP-Completude: Tratamento

Análise de Algoritmos – Ciência da Computação

Prof. Daniel Saad Nogueira Nunes

Instituto Federal de Brasília, Câmpus Taguatinga

Sumário

Tratamento de Problemas $\mathcal{NP}$

- A literatura denomina os problemas que estão em $\mathcal{NP}$ como intratáveis.
- Na prática não é bem assim ...
- Mesmo se um problema for $\mathcal{NPC}$ ainda há esperança.

Tratamento de Problemas $\mathcal{NP}$

- Se a entrada for suficientemente pequena, algoritmos força-bruta podem fornecer uma resposta exata em tempo hábil.
- Casos específicas de problemas $\mathcal{NPC}$ podem ser resolvidas em tempo polinomial.
  - Maior caminho em DAGs.
  - Coloração de vértices quando há apenas duas cores.
- Algoritmos heurísticos: utilizam técnicas heurísticas que sacrificam a qualidade de resposta em prol de uma complexidade menor. Geralmente não garantem uma distância mínima da solução ótima.
- Algoritmos aproximados: possuem uma complexidade viável e garantem uma qualidade da resposta próxima da ideal.
- Focaremos em Algoritmos Aproximados.

Sumário

Sumário

Parâmetros de Aproximação

Definição (Parâmetro de Aproximação)

Dizemos que um algoritmo para um determinado problema possui um parâmetro de aproximação ρ se, para qualquer entrada de tamanho n, o custo C produzido pela solução está a um fator $\rho(n)$ do custo C^* da solução ótima.

Chamamos o algoritmo de $\rho(n)$-algoritmo de aproximação.

Parâmetros de Aproximação

- Matematicamente, a definição anterior pode ser expressa como:

$$\max\left\{\frac{C}{C^*}, \frac{C^*}{C}\right\} \leq \rho(n)$$

- Funciona independentemente do problema ser de maximização ou minimização.
- Note que um algoritmo exato de acordo com esta definição sempre possui parâmetro de aproximação igual a 1.
- Temos $1 \leq \rho(n)$.

Algoritmos Aproximados

- Veremos agora alguns problemas em $\mathcal{NPC}$ que admitem um $\rho(n)$-algoritmo de aproximação.

Sumário

Cobertura de Vértices

- Vimos anteriormente que o problema da Cobertura de Vértices (VC) consiste em encontrar um conjunto $V' \subseteq V$, tal que $|V'| \leq k$ e toda aresta incide nos vértices da cobertura.
- O problema de otimização correspondente, VC-OPTM, consiste em encontrar a cobertura mínima, ou seja, aquela com o menor número de vértices possível.

Cobertura de Vértices

- Sabemos da dificuldade do problema, já que `VC` está em $\mathcal{NPC}$. Assim `VC-OPTM` é pelo menos tão difícil quanto.
- Atacaremos ele utilizando um algoritmo aproximado.

VC-OPTM: Algoritmo Aproximado

Algorithm 1: APPROX-VERTEX-COVER(G)

Input: G

Output: Cobertura aproximada de G

$C \leftarrow \emptyset$

$E' \leftarrow G.E$

while $E' \neq \emptyset$ **do**

  // Escolha de uma aresta arbitrária

  Seja (u, v) uma aresta arbitrária de E'

  // Inserção de u e v na cobertura aproximada

  $C \leftarrow C \cup \{u\} \cup \{v\}$

  // Remoção de todas as arestas incidentes nos nós incluidos na cobertura

  Remova de E' todas as arestas incidentes em u ou v

// Retorna a cobertura aproximada

return C

VC-OPTM: Algoritmo Aproximado

- Claramente o algoritmo leva tempo $O(|V|^2)$ se utilizada uma matriz de adjacências.
- Mostraremos agora que o algoritmo é de fato um 2-algoritmo de aproximação.

VC-OPTM: Algoritmo Aproximado

Theorem

APPROX-VERTEX-COVER *é um* 2*-algoritmo de aproximação de tempo polinomial.*

`VC-OPTM`: Algoritmo Aproximado

Demonstração

O conjunto C retornado pelo algoritmo certamente é uma cobertura, uma vez que o algoritmo prossegue até remover todas as arestas de $G.E$ e portanto, todas estão cobertas por C.
Olhemos para a escolha das arestas feita na linha 4 do algoritmo.
Seja A o conjunto de arestas escolhido pelo algoritmo. Uma cobertura ótima C^* precisa incluir pelo menos uma das extremidades de cada $e \in A$.
Além disso, podemos concluir que nenhum vértice de uma extremidade de $e \in A$ se liga a outro vértice que está em uma das extremidades de $e' \in A$, devido à linha 6 do algoritmo.

VC-OPTM: Algoritmo Aproximado

Demonstração

Conseguimos concluir então que:

$$|C^*| \geq |A|$$

Ao mesmo tempo, cada execução da linha 5 pega as duas extremidades de cada aresta de A, logo podemos concluir que:

$$C = 2|A|$$

VC-OPTM: Algoritmo Aproximado

Demonstração

Combinando as duas desigualdades, temos:

$$\begin{aligned} |C| &= 2|A| \\ &\leq 2|C^*| \end{aligned}$$

$\square$

Caixeiro Viajante

Definição (TSP-OPTM)

A versão de otimização do problema do Caixeiro Viajante (TSP) consiste em, dado um grafo $G = (V, E)$ com uma função $c : E \to \mathbb{R}^+$, determinar um ciclo hamiltoniano G de menor custo, ou seja, um ciclo que visita cada cidade apenas uma vez e volta na cidade original com menor custo possível.

- **Entrada**: $G = (V, E)$.
- **Saída**: o valor do ciclo hamiltoniano com menor custo possível.

Caixeiro Viajante

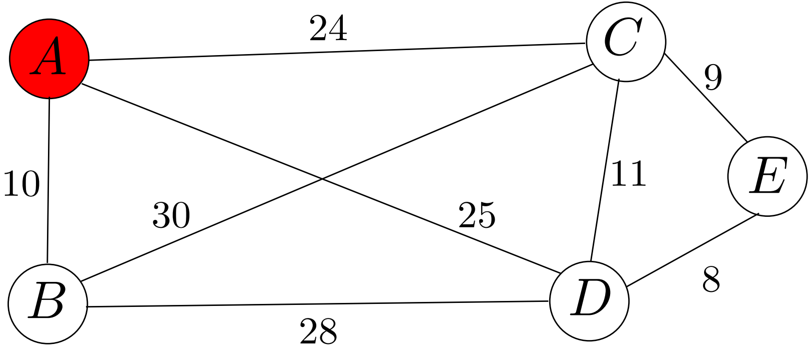
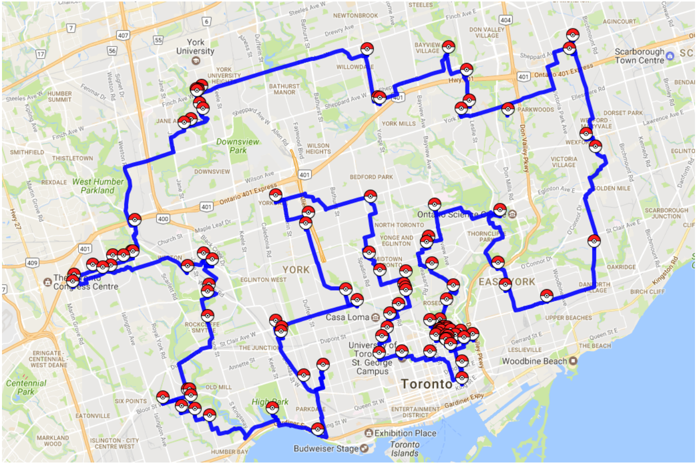
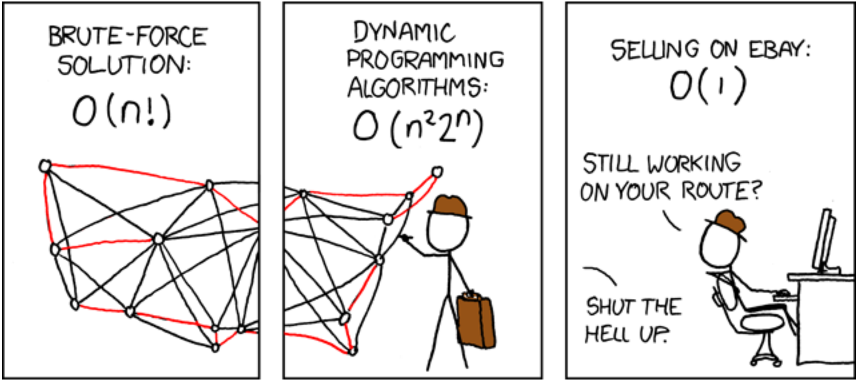


Figura: Qual a resposta?

Caixeiro Viajante



Caixeiro Viajante



Caixeiro Viajante

- Este é um problema difícil, uma vez que a versão de decisão do problema, `TSP`, está em $\mathcal{NPC}$.
- Como atacá-lo através de algoritmos aproximados?
- De fato, apenas a versão do `TSP-OPTM` que obedece a desigualdade triangular possui soluções aproximadas conhecidas.

Desigualdade Triangular

Definição (Desiguldade Triangular para Grafos)

Uma função de custo c satisfaz a desigualdade triangular quando, para quaisquer vértices u, v, $w \in V$, temos:

$$c(u, w) \leq c(u, v) + c(v, w)$$

Ou seja, não é mais custoso ir diretamente a um outro nó do que tentar pegar um atalho através de um terceiro.

Desigualdade Triangular

- Apesar de limitar a apenas grafos que obedecem a desigualdade triangular, vários grafos em problemas reais possuem esta propriedade.
- A distância euclidiana satisfaz a propriedade de distância triangular, logo qualquer grafo que seja modelado como pontos em um plano cartesiano também possui tal propriedade.

TSP-OPTM: Algoritmo Aproximado

- Mostraremos agora como projetar um algoritmo com parâmetro de aproximação 2 para TSP-OPTM.

TSP-OPTM: Algoritmo Aproximado

Algorithm 2: $\textsc{Approx-TSP-Tour}(G, c, r)$

Input: G, c, r

Output: Solução aproximada para TSP

```
// Computa a árvore geradora mínima de G com raiz em r
T ← MINIMUM-SPANNING-TREE(G, c, r)
// Executa a busca pré-ordem em T e retorna a lista dos
   nós
H ← PREORDER(T, r)
// Remove nós duplicados do passeio
REMOVE-DUPLICATE(H)
// Retorna o ciclo hamiltoniano
return (H)
```

TSP-OPTM: Algoritmo Aproximado

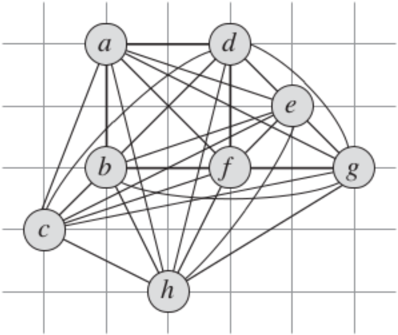


Figura: Grafo Original.

TSP-OPTM: Algoritmo Aproximado

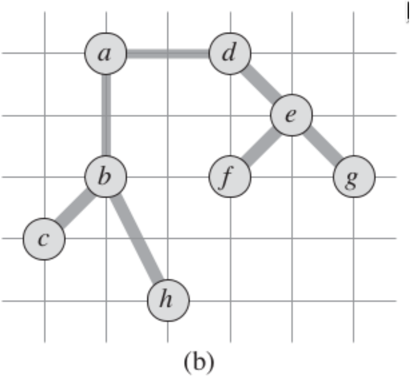


Figura: Árvore Espalhada Mínima.

TSP-OPTM: Algoritmo Aproximado

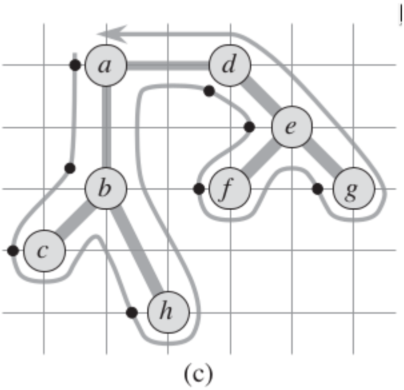


Figura: Percurso em Pré-Ordem.

TSP-OPTM: Algoritmo Aproximado

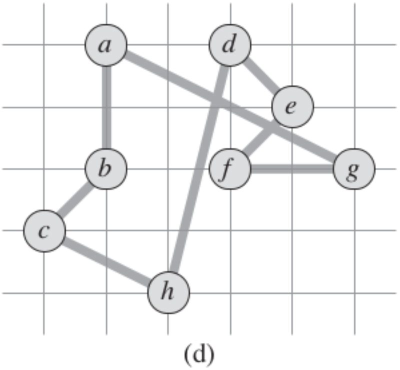


Figura: Obtenção do Circuito Hamiltoniano.

TSP-OPTM: Algoritmo Aproximado

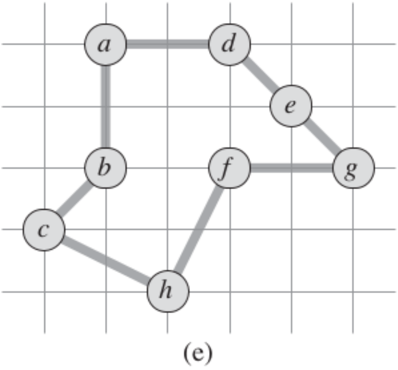


Figura: Solução Ótima.

TSP-OPTM: Algoritmo Aproximado

Theorem

APPROX-TSP-TOUR *é um* 2*-algoritmo de aproximação para o* TSP-OPTM *para grafos que obedecem a desigualdade triangular.*

TSP-OPTM: Algoritmo Aproximado

Demonstração

Primeiramente, é fácil ver que o algoritmo APPROX-TSP-TOUR executa em tempo polinomial dominado pela construção da árvore geradora mínima.
Precisamos mostrar agora que a solução obtida está no máximo a um fator de 2 da solução ótima.

TSP-OPTM: Algoritmo Aproximado

Demonstração

Seja $c(T)$ o custo da árvore geradora mínima obtida e $c(H^*)$ a melhor resposta possível. Claramente temos:

$$c(T) \leq c(H^*)$$

TSP-OPTM: Algoritmo Aproximado

Demonstração

Durante o percurso em T, foi obtido uma lista de vértices W que representa o passeio na árvore geradora mínima. Claramente, como o passeio atravessa cada aresta duas vezes, temos:

$$c(W) = 2c(T)$$

Logo:

$$c(W) \leq 2c(H^*)$$

TSP-OPTM: Algoritmo Aproximado

Demonstração

No entanto, o passeio não necessariamente corresponde a um ciclo Hamiltoniano. Ao desprezar os nós repetidos no passeio e utilizando a desigualdade triangular, podemos obter um ciclo hamiltoniano H, que com certeza possui:

$$c(H) \leq C(W)$$

Observando as desigualdades, é fácil concluir que:

$$c(H) \leq 2c(H^\star)$$

□