Tratamento de Problemas NP-Difíces
Branch-and-Bound

Cid C. Souza
Eduardo C. Xavier

Instituto de Computação/Unicamp

9 de setembro de 2013

Introdução

- Aplicado a problemas onde se quer otimizar uma *função objetivo*.
- Exploração do espaço de estados: todos os filhos de um nó da árvore de espaço de estados são gerados ao mesmo tempo.
- Em cada nó da árvore, a *função classificadora* estima o melhor valor da função objetivo no subespaço das soluções representadas por aquele nó.
- Os nós são *amadurecidos* por:
  1. <u>inviabilidade</u> (não satisfazer as restrições implícitas);
  2. <u>limitante</u> (função classificadora indica que ótimo não pode ser atingido naquela subárvore)
  3. <u>otimalidade</u> (ótimo da subárvore já foi encontrado).

Conceitos

Inicialmente assumimos problemas de minimização.

- Uma estrutura de dados é utilizada para armazenar nós ativos:
  - ADD(X): Adiciona o nó X na estrutura.
  - NEXT(): Retorna o próximo nó da estrutura ou indicação de que está vazia.
- Dado um nó X a função de custo z deste nó é:
  - Se X é uma solução inviável $z(X) = \infty$.
  - Se X é uma solução viável $z(X)$ é igual ao custo da solução.
  - Se X é parte de uma solução $z(X)$ é igual ao custo da melhor solução que pode ser gerada a partir de X.

Conceitos

- A função z é difícil de ser calculada.
- Na estratégia BaB utilizamos uma função $\underline{z}$ que aproxima z.
  - Para qualquer nó X temos que $\underline{z}(X) \leq z(X)$.
  - $\underline{z}$ é um **limitante inferior** para z.
  - Se X é uma solução completa então $\underline{z}(X) = z(X)$.
- Também é utilizada uma função $\overline{z}$
  - $\overline{z}(X)$: valor máximo que uma solução tendo a parte X terá.
  - $\overline{z}$ é um **limitante superior**.
  - $z(X) \leq \overline{z}(X)$.
- Denotamos por U o menor valor de $\overline{z}(X)$ dentre nós ativos X.

Conceitos

Bound (considerando problemas de minimização):

- Um nó ativo X será avaliado somente se $\underline{z}(X) \leq U$.
- Caso contrário o nó é amadurecido (descartado).
- Quando um nó corrente é tal que $\overline{z}(X) < U$, atualizamos o valor de U.

Algoritmo

Considerando problemas de **minimização**.

$\textsc{BaB}$ (T)

  NósAtivos.ADD(T); MelhorSol $\leftarrow \{\}$; $U \leftarrow \infty$;

  Enquanto (NósAtivos não está vazia) **faça**

    $E \leftarrow$ NósAtivos.NEXT();

    Para cada filho X de E **faça**

      Se X é uma solução viável: $\underline{z}(X) = \overline{z}(X) = z(X)$

      Se X é inviável: $\underline{z}(X) = \overline{z}(X) = \infty$

      Se X é solução parcial calcula $\underline{z}(X)$ e $\overline{z}(X)$
          por heurísticas.

      Se $\underline{z}(X) \leq U$ **então**

        Se X é uma solução completa **então**

          MelhorSol $\leftarrow X$

          $U \leftarrow z(X)$

        Senão

          NósAtivos.ADD(X) e $U \leftarrow \min\{U, \overline{z}(X)\}$

  Fim-Enquanto

Algoritmo

- Se a estrutura *NósAtivos* for uma Fila, a busca se dará em largura.
- Se a estrutura *NósAtivos* for uma Pilha, a busca se dará em profundidade.
- É também comum implementar a estrutura como um Heap de mínimo.
  - Estratégia do melhor limitante (*best bound*).
  - Uma busca gulosa pelos nós mais promissores.
  - No caso de minimização escolher nó com menor $\underline{z}(X)$.
- Vamos assumir o uso de um Heap com estratégia *best bound*.

Algoritmo

Considerando problema de **minimização** .

BaB-Best (T)

  NósAtivos.ADD(T); $U \leftarrow \infty$;

  Enquanto (NósAtivos não está vazia) **faça**

    $E \leftarrow$ NósAtivos.NEXT();

    Se E for uma solução completa então **retorne** E

    Para cada filho X de E **faça**

      Se X é uma solução viável: $\underline{z}(X) = \overline{z}(X) = z(X)$

      Se X é inviável: $\underline{z}(X) = \overline{z}(X) = \infty$

      Se X é solução parcial calcula $\underline{z}(X)$ e $\overline{z}(X)$
          por heurísticas.

      Se $\underline{z}(X) \leq U$ **então**
          NósAtivos.ADD(X) e $U \leftarrow \min\{U, \overline{z}(X)\}$

  Fim-Enquanto

Teorema

A solução retornada por BaB-Best *é uma solução ótima.*

Prova. Seja E o nó retornado pelo algoritmo. Todos os nós descartados possuem custo maior que U e portanto maior do que E.

No instante que E é retornado da estrutura temos que $z(E) = \underline{z}(E) \leq \underline{z}(X)$ para qualquer nó ativo X. Para qualquer nó Y que ainda será gerado como descendente de algum nó X temos $z(Y) \geq \underline{z}(X) \geq \underline{z}(E)$.

Portanto E é uma solução ótima. □

Traveling Salesman Problem **(TSP)**

- Temos um grafo não direcionado $G = (V, E)$ com custos positivos nas arestas.
- Achar um ciclo de custo mínimo passando por todos os vértices.
- Custos dados por uma matriz de adjacência.

Traveling Salesman Problem **(TSP)**

- Representaremos um nó por uma tupla $X = (x_1 = 1, x_2, \ldots, x_n)$.
  - $x_i \in \{2, 3, \ldots, n\}$ indica quem é o *i*-ésimo vértice no ciclo.
- O ciclo parcial no nó X é $C_X = (x_1 = 1, x_2 = v_2, \ldots, x_k = v_k)$.
- Podemos usar a função classificadora $\underline{z}(X)$ como:
  - Para cada $v \in V \setminus V(C_X)$ ache duas arestas incidentes de peso mínimo e_v^1, e e_v^2.
  - Ache aresta de peso mínimo e_1 incidente a 1 e e_k incidente a v_k.
  - $\underline{z}(X) = \sum_{e \in E(C_X)} c(e) + \frac{e_1 + e_k + \sum_{v \in V \setminus V(C_X)} (e_v^1 + e_v^2)}{2}$

Traveling Salesman Problem **(TSP)**

Considere a matriz de custos: $\begin{pmatrix} 0 & 7 & 12 & 8 & 11 \\ 7 & 0 & 13 & 7 & 10 \\ 12 & 13 & 0 & 9 & 12 \\ 8 & 7 & 9 & 0 & 10 \\ 11 & 10 & 12 & 10 & 0 \end{pmatrix}$

- Inicialmente:

$$\underline{z}(1) = [(7+8)+(7+7)+(9+12)+(7+8)+(10+10)]/2 = 84/2 = 42.5$$

- Após inserir o vértice 2:

$$\underline{z}(1,2) = 7 + [8 + 7 + (9 + 12) + (7 + 8) + (10 + 10)]/2 = 42.5$$

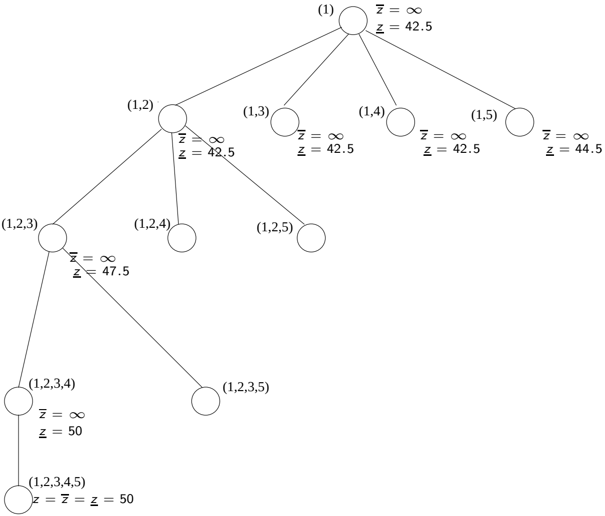
- Após inserir o vértice 5:

$$\underline{z}(1,2,5) = 7 + 10 + [8 + 10 + (9 + 12) + (7 + 8)]/2 = 44$$

mas note que no vértice 4 o custo mínimo de 7 é para o vértice 2 (é inviável). Podemos melhorar para

$$\underline{z}(1,2,5) = 7 + 10 + [8 + 10 + (9 + 12) + (9 + 8)]/2 = 45$$

Traveling Salesman Problem (TSP)



Traveling Salesman Problem **(TSP)**

- Podemos calcular $\overline{z}(X)$ com um idéia parecida:
  - Para cada vértice ache as duas arestas de maior peso incidente.
  - Ache a aresta de maior peso incidente em 1 e em v_k.
  - Some os custos destas arestas e divida por 2 e some ao custo de C_X.
- Assim como antes deve-se considerar apenas arestas válidas, ou seja, que não incidam em vértices no "meio" de C_X.