

PUC-Rio
Departamento de Informática
Prof. Marcus Vinicius S. Poggi de Aragão
Horário: 3as-feiras de 13 às 16 horas - Sala 154L
2 de junho de 2009
Data da Entrega: 13 de julho de 2009
Período: 2009.1

PROJETO E ANÁLISE DE ALGORITMOS (INF 2926)

2º Trabalho de Implementação

Descrição

O objetivo do 2º Trabalho é a implementação e a avaliação experimental algoritmos para um problema NP-difícil. Isto é, um problema de otimização cuja versão de decisão é um problema NP-completo. O objetivo é obter as soluções ótimas de instâncias do problema proposto e provar que estas soluções são realmente ótimas. Assim, um algoritmo de *branch-and-bound* deverá ser implementado. O problema sobre o qual deve-se aplicar o algoritmo de *branch-and-bound* segue:

- **Problema da Árvore Geradora com Restrições de Capacidade (CAP-MST):**

Dado um grafo $G = (V \cup \{r\}, E)$, custos $c(i, j) \geq 0$ associados às arestas (i, j) em E , demandas $q(v)$ associadas aos vértices v em V , e uma capacidade Q . Deseja-se encontrar uma árvore geradora de G , cujas sub-árvores enraizadas no vértice r possuam a soma das demandas dos seus vértices inferiores ou iguais à Q , cujo custo total de suas arestas seja **mínimo**.

- As instâncias estão no link CAP-MST da página do curso.
- **Aplicação:** Para aqueles que gostariam de conhecer um contexto onde este problema é importante, considere que no vértice r estão situados roteadores onde cada um dos seus *slots* tem uma capacidade, C MB/s por exemplo. Assim, a soma da demanda de todos os clientes na árvore conectada em cada *slot* não pode ultrapassar a capacidade de tráfego do *slot*, no caso C MB/s.
- A apresentação do seu algoritmo deve conter uma descrição de cada um dos elementos em um *branch-and-bound*, são eles:
 - Critério de particionamento do espaço de soluções. Sugere-se que o critério utilizado seja o de particionar em dois conjuntos: um onde a árvore geradora contém obrigatoriamente uma dada aresta; e outro onde esta aresta não está na árvore.
 - Um critério para percorrer os subconjuntos do espaço de soluções. Para facilitar a implementação, sugere-se o critério de busca em profundidade.

- Uma relaxação do problema (uma função que gera sempre um valor inferior ou igual ao da solução ótima para o subconjunto de soluções considerado). Neste item, mostre como a sua relaxação é modificada quando se considera apenas um subconjunto do espaço de soluções (ou seja, quando um elemento da solução é fixado, neste caso isto corresponde a uma aresta ter que estar obrigatoriamente na árvore geradora ou obrigatoriamente fora dela). Uma relaxação simples é a árvore geradora mínima que não considera a restrição de capacidade.
 - Um método para obter uma “boa” solução viável (que seria a melhor solução conhecida antes de iniciar o *branch-and-bound*). Esta solução pode ser obtida por um método guloso, por exemplo, que inicie com cada vértice em uma sub-árvore e proceda reunindo pares de sub-árvores que reduzem ao máximo o custo da árvore geradora corrente e não violem a capacidade da sub-árvore.
 - Critério de seleção do particionamento. Uma possibilidade é ordenar as arestas pelos seus respectivos custos. Outras ordens podem ser tentadas.
- Apresente sempre a sua melhor solução (lista das arestas com seus respectivos valores) e o seu valor total. Caso o tempo de CPU ultrapasse 1 hora, faça com que seu algoritmo termine imprima a melhor solução encontrada até então. Apresente sempre, também, o valor do menor limite inferior para a solução ótima no momento em que a enumeração foi interrompida. Deverá ser apresentado um relatório sobre as experiências computacionais comentando os resultados obtidos. Este relatório deverá conter:
 - Uma tabela com o valor da melhor solução obtida, indicando se é ótima (provado pelo seu algoritmo) ou não, o tempo de cpu total utilizado na resolução, o valor do limite inferior obtido no nó raiz, e o valor da solução ótima (ou melhor conhecida) que serão fornecidos. A tabela deverá ter uma linha para cada uma das 39 instâncias contidas no arquivo no link;
 - Uma análise dos resultados com relação à complexidade assintótica do algoritmo implementado. Mostrar que o crescimento do tempo é exponencial em função do tamanho da instância;
 - Uma análise separada das diferentes etapas do algoritmo.
 - Os códigos (comentados) devem ser entregues eletronicamente apenas. Um roteiro para o documento a ser entregue segue:
 - Descrever os algoritmos informalmente.
 - Demonstrar o entendimento do algoritmo explicando, em detalhe, o resultado que o algoritmo deve obter e justificá-lo.
 - Explicar a fundamentação do algoritmo e justificar a sua corretude. Apresentar e explicar a complexidade teórica esperada para cada algoritmo.
 - Documente o arquivo contendo o código fonte de modo que cada passo do algoritmo esteja devidamente identificado e deixe claro como este passo é executado.
 - A corretude e a qualidade do código será testada sobre um conjunto de instâncias que será distribuído. O trabalho entregue deve conter:

- Um documento contendo o roteiro de desenvolvimento dos algoritmos (e dos códigos), os itens pedidos acima, comentários e análises sobre a implementação e os testes realizados (papel).
- Envie um e-mail contendo um arquivo . zip com os códigos fonte e os executáveis correspondentes (mudar a extensão de .zip para .zxx) para **poggi@inf.puc-rio.br** com o ASSUNTO (ou SUBJECT) PAA091T2.