

Período: 2009.2

23 de setembro de 2009

Horário: 2as-feiras e 4as-feiras de 15 às 17 horas

ANÁLISE DE ALGORITMOS (INF 1721)

2º Trabalho de Implementação (T2)

Data de Entrega: 21 de dezembro de 2009

- O objetivo do 2º Trabalho é a implementação de um algoritmo de *branch-and-bound* para um problema NP-difícil. Isto é, um problema de otimização cuja versão de decisão é um problema NP-completo.

Este trabalho deve ser feito em grupos de no MÁXIMO 5(CINCO) alunos. Caso contrário não será considerado.

- O problema sobre o qual deve-se aplicar o algoritmo de *branch-and-bound* segue:
 - Problema de Empacotamento em Caixas Unidimensionais (*Bin Packing Problem 1-D*, **BPP-1**): Dado um conjunto de caixas idênticas com capacidade C , em termos de peso, e n objetos com pesos inteiros representados por w_1, w_2, \dots, w_n , determinar o número **mínimo** de caixas para empacotar todos os n objetos.
3 Conjuntos de instâncias para o (BPP-1) podem ser encontrados na página do curso, foram obtidas na link <http://www.wiwi.uni-jena.de/Entscheidung/binpp/index.htm>.

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. Conforme apresentado em aula um critério possível é o de particionar em dois conjuntos: um onde dois objetos ficam na mesma caixa, e outro onde estes ficam em caixas separadas.
- Uma relaxação do problema, isto é, uma função que obtém um valor garantidamente menor ou igual (minimização) ao da solução ótima do subconjunto considerado do espaço de soluções (o subconjunto pode ser, e é no início, o conjunto de todas as soluções possíveis). Neste item, mostre como a sua relaxação é modificada quando se considera apenas um subconjunto do espaço de soluções (ou seja, no critério de particionamento sugerido, quando dois objetos estão obrigatoriamente separados ou quando estão necessariamente juntos numa caixa). A referência abaixo fornece um estudo sobre limites inferiores (relaxações) para o BPP-1:

S.P. Fekete and J. Schepers, “New classes of fast lower bounds for bin packing problems”, *Mathematical Programming* 91 (2001) 1, 11-31

Este artigo pode ser obtido na página de periódicos da CAPES. (Esse artigo tem um pequeno erro na primeira página, qual é ? :-))

- 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.
- Critério de seleção do particionamento a ser feito em cada nó da árvore de busca (ou seja, a critério de escolha dos 2 objetos no particionamento).
- Critério de percorrimento da árvore de busca (sugere-se que seja busca em profundidade, permitindo uma implementação recursiva e mais simples de ser feita).

Apresente sempre a sua melhor solução (lista dos objetos em cada caixa) e o seu valor total (número de caixas utilizadas). Caso o tempo de CPU ultrapasse 1 hora, faça com que seu algoritmo termine imprima a melhor solução encontrada até então.

Deverá ser apresentado um relatório sobre as experiências computacionais comentando os resultados obtidos. Este relatório deverá conter:

- Complexidade do Problema BPP-1 (prova de que é NP-difícil);
- 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 instâncias contidas nos arquivos nos links;
- Uma análise dos resultados com relação à complexidade assintótica do algoritmo implementado. Mostrar 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 correte. 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 correte código será testada sobre o conjunto de instâncias no link acima. 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) AA092T2.