

Armazenamento de Arquivos Grandes em Dvds

MAC5758 - Introdução ao Escalonamento e Aplicações

Viviane Teles de Lucca Maranhão

Instituto de Matemática e Estatística da Universidade de São Paulo

Dezembro de 2009

Sumário

- 1 Introdução
- 2 Objetivos
- 3 Heurísticas
- 4 Exemplo
- 5 Implementação
- 6 Próximas Etapas
- 7 Referências Bibliográficas

O problema

Problema Principal

Dados n arquivos de tamanhos t_1, t_2, \dots, t_n , distribuí-los em x DVDs de tamanho C de modo que x seja o menor possível.

Exemplos de áreas com problemas semelhantes

- Carpintaria;
- Indústrias;
- Emissoras de televisão.

Formulação

Bin-packing

Dadas $L = \langle a_1, a_2, \dots, a_n \rangle$ uma lista com números no intervalo $(0, 1]$, e uma seqüência de *bins* (recipientes) com capacidade unitária B_1, B_2, \dots . Encontrar uma atribuição dos números aos *bins* de modo que em nenhum *bin* a soma dos números atribuídos a ele seja maior que 1, e tal que o número de *bins* utilizados é minimizado.

Objetivos

- Estudar heurísticas de resolução do *Bin-packing*;
- Elaborar um programa em Python que sugira gravações de arquivos passados pelo usuário em uma mídia de tamanho dado.

Terminologia

- Um recipiente *aberto* é aquele no qual podemos inserir itens;
- Um recipiente *fechado* é aquele no qual não podemos mais inserir itens;
- R_A^∞ é a razão assintótica de pior caso
- $R_A^\infty(\alpha)$ é a razão assintótica de pior caso com tamanho limitado dos itens;
- $\bar{R}_A^\infty(F)$ é a razão assintótica esperada para A sob F .

Next-Fit (NF)

Descrição

Mantém apenas um recipiente aberto por vez; quando um novo recipiente precisa ser aberto o anterior é fechado.

Desempenho

- $R_{NF}^{\infty} = 2$
- $R_{NF}^{\infty}(\alpha) = \frac{1}{1-\alpha}$, $\alpha < \frac{1}{2}$
- $\bar{R}_{NF}^{\infty}(U[0, 1]) = \frac{4}{3}$

First-Fit (FF)

Descrição

Insere o item no primeiro recipiente possível, ou abre um novo se não couber nos abertos até o momento.

Desempenho

- $R_{FF}^{\infty} \cong 1.69103$
- Para $\frac{1}{m+1} < \alpha \leq \frac{1}{m}$
 $R_{FF}^{\infty}(\alpha) = 1.7$ se $m = 1$ e $R_{FF}^{\infty}(\alpha) = 1 + \frac{1}{m}$, se $m \geq 2$
- $\bar{R}_{FF}^{\infty}(U[0, 1]) = 1$

Best-Fit (BF)

Descrição

A cada alocação todos os recipientes são avaliados e o que apresentar a menor sobra de espaço após a alocação é selecionado. Assim como nos casos anteriores, um novo recipiente é aberto quando não é possível alocar o item em nenhum dos recipientes anteriores.

Desempenho

- $R_{BF}^{\infty}(\alpha) = R_{FF}^{\infty}(\alpha)$
- $\bar{R}_{FF}^{\infty}(U[0, 1]) = 1$

Worst-Fit (WF), Almost-Worst-Fit (AWF)

Descrição

O WF seleciona o recipiente no qual que resta o maior espaço após a alocação. O AWF aloca o item no recipiente que após a alocação deste apresente a segunda maior sobra de espaço.

Desempenho

- $R_{WF}^{\infty}(\alpha) = R_{NF}^{\infty}(\alpha)$, $0 < \alpha \leq 1$
- $R_{AWF}^{\infty}(\alpha) = R_{FF}^{\infty}(\alpha)$, $0 < \alpha \leq 1$

First-Fit-Decreasing (FFD), Best-Fit-Decreasing (BFD)

Descrição

Ordenam o vetor com os itens em ordem não crescente de tamanho antes de aplicar o FF ou o BF, respectivamente. Pela possibilidade de ordenação prévia garantem melhor desempenho que seus respectivos algoritmos *online*.

Desempenho

- $R_{FFD}^{\infty} = R_{BFD}^{\infty} = \frac{11}{9} = 1.222\dots$

Instância

Vamos considerar a seguinte lista de objetos

$L = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)$ com seus tamanhos
 $S(L) = (\frac{1}{2}, \frac{3}{4}, \frac{3}{8}, \frac{2}{5}, \frac{2}{3}, \frac{1}{8}, \frac{3}{5}, \frac{1}{4})$ mostrada na figura 1.

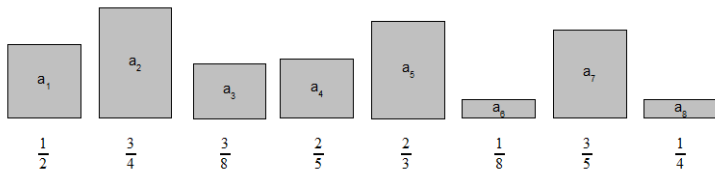
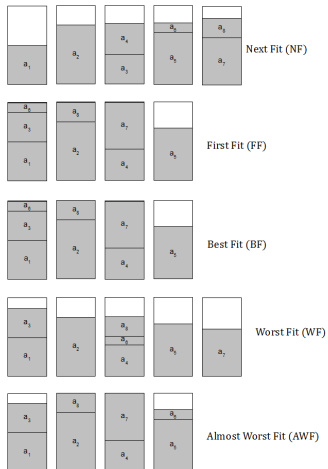
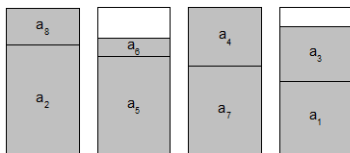


Figura: Exemplo de uma lista de objetos

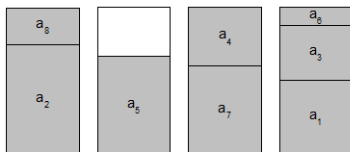
Solução utilizando heurísticas online



Solução utilizando heurísticas offline



First Fit Decreasing (FFD)



Best Fit Decreasing (BFD)

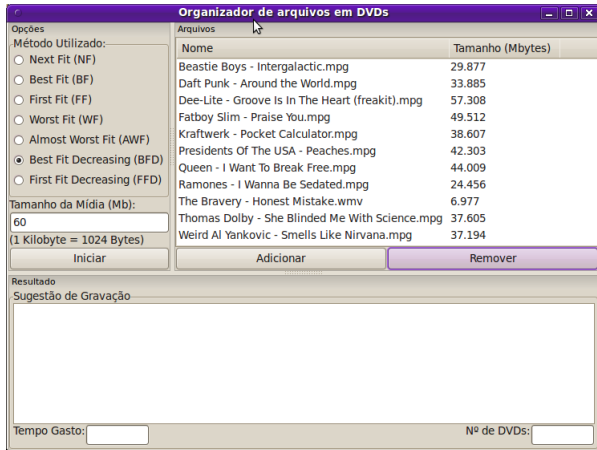
Estrutura do programa

Linguagem utilizada: Python 2.6 com a biblioteca gráfica wxpython versão 2.8.

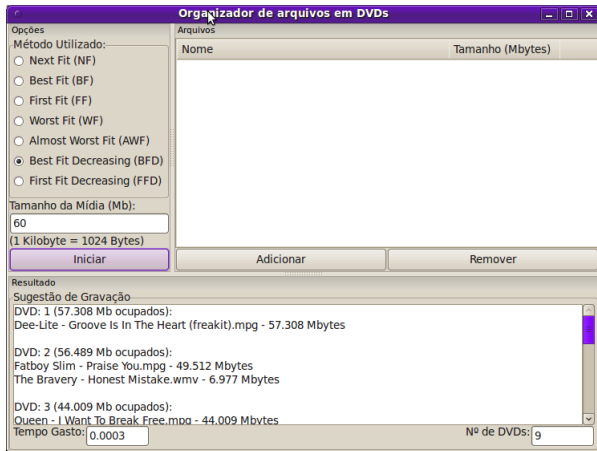
Separamos o programa em dois arquivos:

- **organizador.py** Interface gráfica para o usuário entrar com as informações: tamanho da mídia, lista de arquivos e método desejado para resolução, e que após rodar o método apresenta uma sugestão de gravação.
- **metodos.py** Contém a implementação das heurísticas de empacotamento citadas anteriormente.

Screenshot do programa








Screenshot do programa



Passos a serem realizados

- 1 Teste do programa implementado para diversas instâncias;
- 2 Análise dos resultados obtidos confrontando com os esperados;
- 3 Elaboração das conclusões.

-  Johnson D. S., Near-optimal bin packing algorithms, Massachusetts Institute of Technology (1973)
-  Xavier, E. C., Miyazawa F. K., Algoritmos para Problemas de Empacotamento, Anais do XXVII Congresso da SBC, CTD, XX Concurso de Teses e Dissertações, p1966 - 1973, Rio de Janeiro (2007)
-  Hochbaum, D. , Approximation algoritms for NP-hard problems, PWS Publishing Company, Boston (1995)
-  Wronski, F., Alocação Dinâmica de Tarefas em NoCs Malha com Redução do Consumo de Energia, Universidade Federal do Rio Grande do Sul, Dissertação de Mestrado, Porta Alegre, (2007)
-  Coffman E. G, et. Al. Average Analysis of on-line Bin-packing Algorithms. OR-Seminar, Michaelmas (1996)

Obrigada