Disciplina MC658
Projeto e Análise de Algoritmos III
2.o Semestre de 2015 - Prof.: Flávio Keidi Miyazawa
Instituto de Computação - UNICAMP

Marcelo A. G. dos Santos RA:106140

Laboratório 3 - Atribuição de Propagandas Relacionadas

O problema

A empresa Google deseja disponibilizar uma faixa horizontal de sua tela para colocar propagandas e obter receita com isso. Em uma tela, que pode ser de um computador ou qualquer outro dispositivo, existe uma área limitada C na qual as propagandas podem aparecer. A empresa dispõem de P propagandas que podem ser mostradas, na qual uma propaganda i tem um custo c_i e um valor v_i. A combinação de propagandas i e j gera um custo/valorização que é dado pela matriz W[i][j]. O objetivo do laboratório é então encontrar um conjunto de propagandas que maximize seu lucro e que não ultrapasse a área C. Descrevendo formalmente o problema temos que:

$$\max \sum_{i \in S} v_i + \sum_{i \in S} \sum_{j \in S} w_{i,j}$$

$$\sum_{i \in S} c_i \leq C$$

$$S \subseteq P$$

em que S é o conjunto de propagandas que foram selecionadas.

Backtracking

Para solucionar esse problema sem a utilização de programação linear inteira fizemos uma abordagem de backtracking. Nesse abordagem, cria-se uma árvore binária na qual cada nó represente a decisão de colocar o item i na solução ou não e então expandir esse nó. Tomemos como exemplo a primeira entrada do laboratório representada no grafo abaixo com os primeiros 20 vértices.

Em cada nó temos as seguintes informações: a ordem em que ele foi visitado, a próxima propaganda que será considerada, as propagandas que foram selecionados ou não,o custo e o lucro obtido naquele nó.
Obviamente expandir todos os nós não é a melhor saída, uma vez que para um número de propagandas P, temos 2^P possibilidades. Para podermos podar nós dessa árvore, temos três opções:

1. Criamos uma variavel que indica a melhor solução encontrada até o momento. Se concluirmos que um determinado nó tem um potencial de lucro inferior ao valor de melhor lucro encontrado, não realizamos a expansão desse nó, uma vez que não iremos obter uma solução melhor.
2. Se um determinado nó tem um custo maior ou igual ao permitido, esse nó não precisa ser expandível, já que não podemos acrescentar nenhuma propaganda a mais nessa solução.
3. Se a próxima propaganda a ser considera for maior que o número total de propagandas existentes.

Limite Superior

Para calcularmos a limite superior de um vértice, ou seja, qual a melhor solução que sua expansão pode conseguir, ordenamos as propagandas em ordem decrescente de v_i / c_i.
Em seguida, partimos da próxima propaganda que será considerada na solução e vamos adicionando o seu valor e custo ao lucro e custo obtidos naquele momento. Além dos valores das propagandas, incrementamos nesse limite superior todos os valores positivos da matriz de custo/valorização da nova propaganda que foi adicionada. Quando então atingimos um custo total K tal que a se adicionarmos a próxima propaganda i na solução teremos um valor K igual ou supeior ao limite de custo, adicionamos um lucro parcial dessa propaganda fazendo com que K = C. Por fim, adicionamos todos os valores positivos da matriz de custo/valorização de todos as propagandas que não entraram nessa solução. De forma matemática, calculamos o Limite Superior de um vértice com lucro L, custo K, número N de propagandas e a próxima propaganda i da seguinte forma:

$$
\begin{aligned}
&Limite\ Superior = L \\
&Se\ K <= C\ e\ i < Total\ de\ propagandas \\
&\quad Enquanto\ K + c_i <= C \\
&\quad\quad Limite\ Superior = Limite\ Superior + v_i \\
&\quad\quad K = K + c_i \\
&\quad\quad i = i + 1 \\
&\quad Limite\ Superior = Limite\ Superior + (C - K) * v_i/c_i \\
&\quad i = i + 1 \\
&\quad Enquanto\ i < N: \\
&\quad\quad Limite\ Superior = Limite\ Superior + W[i][j],\ com\ 0 \leq j \leq N\ e \\
&\quad\quad i = i + 1
\end{aligned}
$$

O Algoritmo

Em alto nivel, o algoritmo procede da seguinte maneira:

1. **seleciona_propagandas**(int n, double C, double V[N], double P[N], double w[N][N], int *S, double *UpperBound, long t) {
    a. Salvamos as variaveis de entrada em variaveis globais
    b. Inicializa o cronômetro para não excedermos o máximo valor de processamento permitido dado por t.
    c. Ordenamos as propagandas em ordem decrescente da razão do valor pelo custo
    d. Faz a chamada da função de **backtracking()**
    e. Salva o melhor custo obtido juntamente com as propagandas que foram selecionadas

2. **backtracking**(int i, double custo, double lucro, int include[])
    a. Se obtivemos um lucro maior que o melhor lucro obtido até o momento sem exceder o custo máximo:
        i. atualizamos o lucro máximo obtido
        ii. atualizamos o vetor com as propagandas selecionadas
    b. Verificamos se vamos expandir o nó *i* fazendo a chamada da função **pode_expandir()**
        i. Se for possivel expandir, fazemos a chamada de backtracking com o *i+1* quando *i* é selecionado (e recalculamos o lucro e custo) e quando *i* não é selecionado

3. **pode_expandir**(int i,double custo, double lucro)
    a. Se o tempo de processamento máximo foi atingido OU o custo >= custo máximo OU não temos mais propagandas para selecionar (*i >= número de propagandas)*, finalizamos o processamento.
    b. Caso contrário, calculamos o Limite Superior da forma que foi mostrada previamente
    c. Se Limite Superior foir maior ou igual ao lucro máximo obtido, iremos realizar a expansão.

Resultado

Para testarmos o algoritmo, utilizamos a entrada disponivel como padrão e entradas criadas com n variando entre 10 e 40. O resultado da entrada padrão:

Teste	Propagandas Selecionadas	Melhores Propagandas	Custo Obtido	Custo Máximo	Lucro Obtido	Lucro Máximo
1	1 0 0 0 0 0 0 0 1 0	1 0 0 0 0 0 0 0 1 0	97.959	100	197.590	197.590
2	1 1 0 0 0 0 0 1 0 0	1 1 0 0 0 0 0 1 0 0	90.143	100	188.208	188.208
3	0 0 0 1 0 0 1 0 0 1	0 0 0 1 0 0 1 0 0 1	99.869	100	184.541	184.541
4	0 0 0 0 0 0 1 1 1 0	0 0 0 0 0 0 1 1 1 0	99.366	100	209.221	209.222
5	1 0 0 0 0 1 0 0 1 0	1 0 0 0 0 1 0 0 1 0	91.767	100	169.646	169.645
6	0 1 1 0 0 0 0 0 1 0	0 1 1 0 0 0 0 0 1 0	98.076	100	196.025	196.025
7	0 1 0 0 0 1 1 0 0 0	0 1 0 0 0 1 1 0 0 0	96.074	100	206.600	206.600
8	0 1 0 1 1 0 0 0 0 0	0 1 0 1 1 0 0 0 0 0	92.724	100	183.633	183.632
9	1 1 0 0 1 0 0 0 1 0	1 1 0 0 1 0 0 0 1 0	99.514	100	165.057	165.059
10	1 0 0 0 1 1 0 1 0 0	1 0 0 0 1 1 0 1 0 0	95.949	100	165.744	165.745
11	0 0 1 0 0 0 1 0 0 0	0 0 1 0 0 0 1 0 0 0	93.789	100	185.516	185.516
12	1 1 0 0 0 0 1 1 0 0	1 1 0 0 0 0 1 1 0 0	99.727	100	217.903	217.903
13	0 0 0 1 1 0 0 1 0 0	0 0 0 1 1 0 0 1 0 0	97.402	100	190.850	190.849
14	0 0 1 0 0 0 1 0 0 1	0 0 1 0 0 0 1 0 0 1	96.880	100	194.661	194.661
15	0 0 0 0 1 1 0 0 0 1	0 0 0 0 1 1 0 0 0 1	92.738	100	184.232	184.232
16	1 0 0 0 1 0 0 1 0 0	1 0 0 0 1 0 0 1 0 0	99.697	100	185.363	185.362
17	0 0 0 0 1 0 1 1 0 0	0 0 0 0 1 0 1 1 0 0	93.045	100	179.297	179.296
18	0 0 0 0 1 0 1 0 0 1	0 0 0 0 1 0 1 0 0 1	99.738	100	202.070	202.069
19	1 0 0 1 1 0 0 0 0 0	1 0 0 1 1 0 0 0 0 0	99.841	100	172.104	172.104
20	1 0 0 0 0 0 0 0 1 1	1 0 0 0 0 0 0 0 1 1	96.551	100	187.686	187.685

Apesar de termos conseguido selecionar as melhores propagandas ainda assim obtivemos uma diferença entre o resultado esperado com o que foi obtido, fato devido ao arredondamento de ponto flutuante. Com relação ao número de nós criados, obtivemos uma média, para a entrada padrão de 84.7, ou seja, uma média de 4% de nós criados do total de possibilidades, com um máximo de 163 nós e um mínimo de 27. Com relação as outras entradas, o número de nós criados foi inferior a 1% do total de possibilidades, com eficiência de 100%. Nenhum dos testes excedeu o tempo limite de 10s de processamento máximo, comprovando ainda mais a eficiência da abordagem tomada.