Laboratório 2 de MC658

Nome	**RA**
Mariane Previde	121192
Sabrina Beck Angelini	157240

Descrição dos Algoritmos

O problema apresentado é uma variação do *problema da mochila*.

Backtracking

O algoritmo *backtracking* implementado é recursivo com cada chamada recursiva representando um usuário a ser colocado na largura de banda total disponível. Cada chamada verifica se a solução calculada não ultrapassa o limite de largura de banda total considerando a distância *d* entre diferentes classes de usuários presentes na solução sendo calculada. Cada chamada testa as duas possibilidades: uma solução com e uma sem o i-ésimo usuário e seleciona a de maior lucro para a empresa, ou seja, aquela cuja soma de pagamentos dos usuários é máxima.

Como um algoritmo de *backtracking* comum, assim que uma solução apresenta problemas com a restrição de largura de banda total, ela é abandonada junto com seu trecho na árvore de possíveis soluções.

Branch and Bound

O algoritmo *branch and bound* desenvolvido usa como limitante superior da solução a solução do problema da mochila com itens fracionados, ou seja, o algoritmo calcula a taxa lucro/peso para cada item (usuário, no nosso caso) e enche a mochila (largura de banda total, no nosso caso) com os itens de taxa mais alta mais uma parte fracionária do último item utilizado, o lucro total dessa solução é utilizado como limite superior para cada ramo da árvore de soluções.

No cálculo do limitante superior, o algoritmo leva em conta as classes já adicionadas na solução até o nó cujo limitante está sendo calculado, ou seja, considera os espaços *d* de distancia entre os itens já no cálculo do limitante superior.

O algoritmo percorre a árvore de soluções utilizando uma fila de prioridade que dá prioridade aos nós da árvore com maior limitante superior e poda os ramos cujo limitante superior é menor que a melhor solução já encontrada, pois o algoritmo não precisa perder tempo com esses ramos uma vez que esses ramos não são promissores para achar uma solução melhor.

Comparações Computacionais

Resultados

A tabela a seguir mostra dados sobre a execução dos algoritmos acima descritos usando como entrada os arquivos fornecidos no enunciado do laboratório.

Arquivo	Tempo Limite	Algoritmo	Valor da Solução	Tempo de Execução	Solução ótima?	Parou por tempo?
5_2_20.in	10s	Backtracking	17	0.000138s	Sim	Não
5_2_20.in	10s	Branch n Bound	17	0.00035s	Sim	Não
100_5_500.in	10s	Backtracking	398	10.0001s	Não	Sim
100_5_500.in	10s	Branch n Bound	757	7.16629s	Sim	Não
1000_50_800.in	10s	Backtracking	556	10.0039s	Não	Sim
1000_50_800.in	10s	Branch n Bound	1283	10s	Não	Sim

Tabela 1. Resultados e métricas de execução dos algoritmos backtracking e branch and bound paras os arquivos de entrada fornecidos.

Assumimos como solução ótima para cada entrada os dados a seguir que foram enviados por e-mail para a turma:

Arquivo	Solução Ótima
5_2_20.in	17
100_5_500.in	763
1000_50_800.in	2177

Tabela 2. Solução ótima para cada arquivo de entrada fornecido

Conclusão

O Backtracking é um algoritmo que possibilita achar a solução exata para o problema ao analisar praticamente todas as soluções possíveis para ele, ignorando apenas aquelas que não atendem às restrições dadas. Por isso possui uma complexidade muito alta, no nosso caso

utilizamos um timeout de 10s para todas as entradas, pois para casos de entradas muito grandes o tempo de execução seria muito longo.

Enquanto que o Branch n Bound que é um algoritmo otimizado, faz podas da árvore de possíveis soluções através de escolhas estratégias de modo a obter uma aproximação do valor esperado de forma mais rápido, o que é possível notar na tabela 1. A tabela 1 mostra como o algoritmo Branch n Bound terminou sua execução mais rápido do que o Backtracking para os arquivos de 5 e 100 usuários, além disso, as soluções do Branch & Bound são, em geral, muito melhores que a do Backtracking, justamente pelo fato do Branch & Bound conseguir uma solução melhor em menos tempo.

Mesmo assim, o problema é um problema NP-completo, ou seja, não há solução de tempo polinomial para esse problema, vemos isso na entrada grande de 1000 usuários nas duas últimas linhas da tabela 1, ambos os algoritmos deram timeout e não chegaram muito perto da solução ótima. Mais uma vez, nessa mesma entrada de 1000 usuários, vemos que o Branch and Bound é o que chegou mais perto da solução ótima.