Projeto de MC548: Problema da Mochila com Restrições de Conflito

Prof. Flávio K. Miyazawa - 1s2010

1 Introdução

O projeto pode ser feito por um ou por dois participantes e consiste em uma implementação em C/C++ (compilável em linux/gcc por Makefile e linha de comando) de um programa para o Problema da Mochila com Restrições de Conflito. Deverão ser entregues por email, um arquivo .tgz contendo todos os fontes do programa e os diretórios dos pacotes utilizados para a compilação do mesmo. Além disso, deve ser entregue também um relatório. Este projeto pode ser feito tanto pelos alunos da turma MC548A como MC548#.

2 Definição do problema

Um leiloeiro tem n objetos para leiloar, onde os objetos sao dados pelo conjunto $N = \{0, ..., n-1\}$. Há m pessoas interessadas nos objetos, onde as pessoas são dadas pelo conjunto $M = \{0, ..., m-1\}$. Cada pessoa está interessada em exatamente um subconjunto de objetos. A pessoa i está interessada no conjunto $S_i \subseteq N$ e dá um lance de valor p_i pelo conjunto todo. Neste leilão a pessoa está interessada em apenas este conjunto e não tem interesse em pegar menos objetos que os de interesse. Naturalmente um objeto não pode ser vendido para duas pessoas diferentes. Além disso, pode ser que alguns objetos não sejam vendidos para ninguém.

O problema consiste em definir as pessoas que vencem o leilão. Uma pessoa i é um vencedor do leilão, se ela obtém todos os objetos em S_i e com isso paga p_i ao leiloeiro por todos os objetos que está levando. Os perdedores não levam nada nem pagam nada.

O leiloeiro recebe todos os lances e só depois escolhe os vencedores. Como ele quer maximizar o valor da venda, ele deve escolher os vencedores de maneira que o valor total pago a ele seja máximo.

3 Datas e Prazo

O prazo de entrega é 30 de junho as 9hs por e-mail.

Quem entregar até 24 horas depois, terá sua nota multiplicada por 0,8. Quem entregar de 24 até 48 horas depois, terá sua nota multiplicada por 0,6. Os demais terão nota 0,0 no projeto.

4 Sobre estratégias de resolução e tamanho do grupo

Para que você considere as estratégias de resolução do trabalho, observe os seguintes pontos:

E um algoritmo exato contendo pelo menos uma heurística simples.

H conjunto de duas ou mais heurísticas.

M uma metaheurística.

- Se você for fazer o projeto sozinho, poderá escolher um dos itens acima. Se você for fazer o projeto em dupla, deve necessariamente escolher o algoritmo exato (E) e um dos outros dois, i.e., ou um conjunto de heurísticas (H) ou uma metaheurística (M).
- Ao escolher a implementação do exato (E), a heurística simples deve ser usada para podar o processo de enumeração do algoritmo exato.
- Excepcionalmente o item H pode ser de apenas uma heurística se a mesma envolver grande trabalho de implementação. Para isso, justifique o algoritmo e sua implementação e estruturas de dados para o professor. A vantagem de fazer em dupla pode estar na obtenção de soluções melhores, uma vez que as implementações de dois itens podem se complementar na busca de soluções melhores (veja sobre bônus mais abaixo). Por exemplo, se o grupo fez o item (E) e o item (H), então as heurísticas podem ser usada dentro do algoritmo exato e provavelmente o algoritmo exato resolverá instancias maiores ou em menos tempo.

5 Sobre os programas

Todos os programas devem ser feitos na linguagem C/C++. Você deve gerar um Makefile para facilitar a compilação do seu programa. Assim, dando um comando make todos os executáveis devem ser gerados.

Seu programa deve ser modularizado, com comentários no próprio fonte.

Nomes dos programas executáveis

O nome dos programas exato, heurísticas e metaheurísticas que serão executados por linha de comando devem ser:

exato para o programa exato

heuristica para o conjunto de heurísticas. Neste caso, este programa deve executar todas as heurísticas rápidas implementadas e devolver a melhor solução.

metaheuristica para a metaheurística implementada.

Parâmetros

Todos os programas executáveis (exato, heuristica, metaheuristica) obedecerão a uma mesma sintaxe para os parâmetros:

`-f` *entrada* `-t` *tempomax* `-o` *saida*

onde *entrada* deve ser o nome do arquivo de entrada, *tempomax* deve ser o tempo máximo de execução em segundos, *saida* é o nome do arquivo de saida onde será gravada a solução. Por exemplo, a execução seguinte:

```
exato -f entrada.txt -t 60 -o saida.txt
```

executa o programa exato, que lê o arquivo de entrada `entrada.txt` e executa durante 60 segundos no máximo e gera a saida produzida no arquivo `saida.txt`.

Caso algum programa atinja o tempo máximo estipulado nos parâmetros, o programa deve gravar a melhor solução encontrada no arquivo saida.txt.

Formato dos arquivos

O arquivo de entrada é um arquivo texto, onde cada linha apresenta números (separados por pelo menos um espaço em branco) e tem a seguinte cara:

- Na primeira linha, são dados o número de objetos n sendo leiloados e o número de pessoas m candidatas a comprar os objetos, ambos números inteiros positivos. Os objetos são identificados pelos números $\{0, \ldots, n-1\}$. Os potenciais compradores são identificados pelos números $\{0, \ldots, m-1\}$.
- Em seguida, aparecem m linhas, uma para cada comprador. A primeira linha é do comprador 0, a seguinte do comprador 1, e assim por diante até os dados do comprador $m-1$. Cada linha de um comprador, é composto por um número real positivo p_i, um inteiro k_i (quantidade de objetos do conjunto S_i) e em seguida k_i objetos.

Exemplo de arquivo:

```
6 5
10.0   5   0 1 2 3 4
20.0   3   1 2 4
35.0   4   1 3 4 5
25.0   3   0 2 4
30.0   2   1 3
```

No exemplo acima, temos 6 objetos $\{0, 1, 2, 3, 4, 5\}$ e 5 potenciais compradores $\{0, 1, 2, 3, 4\}$. O comprador 0 dá um lance de 10.0 reais para obter os objetos $\{0, 1, 2, 3, 4\}$; o comprador 1 dá um lance de 20.0 reais para obter os objetos $\{1, 2, 4\}$; o comprador 2 dá um lance de 35.0 reais para obter os objetos $\{1, 3, 4, 5\}$; e assim por diante.

O arquivo de saída é um arquivo texto com o seguinte formato. A primeira linha deve conter o número de vencedores do leilão na solução obtida, seguida do tipo de solução obtida dada por um caracter 'E', 'e', 'H', 'M' (exato com solução ótima, exato sem obter solução ótima, heurística, metaheurística), tempo de execução em segundos (número inteiro) e o valor da solução obtida. Nas demais linhas deve se listar os vencedores.

Caso o algoritmo não consiga obter soluções viáveis dentro do tempo estipulado, então a solução gerada deve ser uma instância com quantidade de vencedores igual a 0 (zero).

Possivelmente o tempo de execução do programa será menor que o tempo máximo dado como parâmetro. Nestes casos, a saída deve apresentar o tempo menor, que foi gasto para obter a solução. Se o algoritmo for o exato, então pode ser que ele conseguiu obter a solução ótima dentro do tempo estipulado. Neste caso, o caracter deve ser 'E' (em maiúscula). Caso o algoritmo exato não tenha conseguido obter uma solução ótima dentro do tempo estipulado, então o caracter dever ser dado por 'e' (em minúscula).

Por exemplo, um arquivo de saída para a instância do exemplo de entrada poderia ser:

```
2 E 10 55.0
3
4
```

Neste caso, a primeira linha diz que a solução tem 2 vencedores. Esta solução é composta dos compradores 3 e 4. Note que o comprador 3 deu lance de 25 reais pelos objetos $\{0, 2, 4\}$ e o comprador 4 deu lance de 30 reais pelos objetos $\{1, 3\}$. É uma solução viável pois nã há interseção entre os conjuntos de objetos dos dois

compradores. O programa obteve solução ótima, e por isso foi apresentado com 'E' em vez de 'e'. Para obter a solução ótima, ele gastou 10 segundos de execução (mas o tempo máximo dado como parâmetro pode ter sido bem maior). O valor desta solução é de 55.0 reais (de fato, $30.0 + 25.0$). Note que o objeto 5 não foi vendido para ninguém.

O professor disponibilizará várias instâncias com centenas e até milhares de objetos e potenciais compradores. O objetivo destas instâncias será avaliar o desempenho de cada classe de programa e ver os limites de cada programa para os diferentes tamanhos de instâncias. Assim, não quer dizer que o algoritmo exato deva resolver bem as instâncias grandes, mas ele será comparado com os outros algoritmos exatos implementados pela turma até tamanhos razoáveis para se testar os algoritmos exatos. Naturalmente se espera que heurísticas rápidas possam dar soluções para instâncias grandes.

Pacotes e resolvedores

Os programas poderão usar pacotes disponíveis para resolução de programas lineares e lineares inteiros, como o GLPK (Gnu Linear Programming Kit), disponível livremente. Para aqueles que quiserem usar o resolvedor profissional CPLEX, o professor poderá disponibilizar o acesso a esta biblioteca. Será da responsabilidade do grupo aprender a sintaxe e como usar o resolvedor. Pode ser interessante utilizar o CPLEX, uma vez que ele pode ajudar a gerar programas com mais possibilidades de se obter soluções melhores e com isso obter bonus. Note que o CPLEX já tem uma estrutura toda pronta de programação linear inteira e para isso um algoritmo exato poderá ser feito inserindo a formulação inteira e colocando as heurísticas para serem chamadas durante a execução do *branch and bound* do próprio CPLEX.

6 Avaliação

Os programas que não compilarem ou não derem soluções viáveis para instâncias pequenas terão nota limitada a 3.0. Falta de documentação/comentários ou modularização limitarão a nota do programa a 5.0.

Alguns programas que tiverem bom desempenho sobre um grupo de instâncias, no geral (a ordem será feita pelo professor), poderão ter nota adicional no projeto (neste caso a nota não será truncada). A nota adicional será dada nos seguintes casos:

- O melhor programa exato terá 1 ponto adicional.
- O segundo melhor programa exato terá 0,8 ponto adicional.
- O terceiro melhor programa exato terá 0,5 ponto adicional.
- O melhor conjunto de heurísticas ou metaheurística terá 1 ponto adicional.
- O segundo melhor conjunto de heurísticas ou metaheurística terá 0,8 ponto adicional.
- O terceiro melhor conjunto de heurísticas ou metaheurística terá 0,5 ponto adicional.
- Se um programa ganhar ponto adicional, então este ponto será dado a todos os integrantes do grupo.

Obs.: Programas que tenham uma boa idéia implementada que, juntamente com um relatório mostrando os detalhes e complexidade dos algoritmos implementados, serão bem consideradas pelo professor.

7 Relatório

Além dos códigos fontes, o arquivo `.tgz` (tareado e gzipado) também deve conter um relatório.

Este relatório deve descrever o algoritmo em alto nível e descrito em linguagem algorítmica de maneira que o professor possa entender bem como o(s) algoritmo(s) por trás do programa resolve(m) ou trata(m) o problema.

O relatório deve conter também uma seção de experimentos computacionais, contendo a configuração do computador onde o grupo fez os experimentos (cpu, memória, tipo de processador) e uma tabela contendo os valores que os programas implementados obtiveram para um conjunto de instâncias a ser disponibilizado posteriormente. A tabela deve destacar as soluções onde o programa obteve solução ótima (se souber).

Exercício: Prove que o problema tratado neste projeto é NP-difícil.