Atividade Computacional – Unidade 2

1 Descrição

Implementação em C/C++ de um algoritmo do tipo *branch-and-bound* para o problema da mochila 0-1, de acordo com as regras especificadas neste documento. Recapitulando, o problema, em sua forma geral, é:

$$\max \quad \sum_{i=1}^{n} v_i x_i \tag{1}$$

$$\text{s.a}$$

$$\sum_{i=1}^{n} p_i x_i \leqslant C \tag{2}$$

$$x_i \in \{0,1\},\ i = 1,\ldots,n. \tag{3}$$

Seu algoritmo deve funcionar conforme descrito em videoaula sobre o assunto (vídeo do encontro síncrono do dia 3 de setembro, no Solar). **A relaxação a ser usada deve ser a relaxação linear do problema acima, e esta deve ser resolvida usando o algoritmo guloso[1] mostrado na videoaula.**

Durante a execução, seu código deve exibir as seguintes informações, **a cada 100 nós avaliados**:

- O valor da função objetivo da melhor solução inteira conhecida até o momento.
- O número de nós já avaliados (isto é, que tiveram sua relaxação linear resolvida).
- O número de nós "abertos" (isto é, que ainda não tiveram sua relaxação linear resolvida e aguardam para serem avaliados).
- O valor atual do *gap*.
- Opcionalmente, o tempo decorrido desde o início da execução do algoritmo de B&B.

Você pode querer consultar novamente a aula sobre os parâmetros e o relatório de saída do CPLEX (aula "*Parâmetros do algoritmo branch-and-bound do CPLEX*", do dia 8 de setembro de 2020, no Solar), já que o relatório descrito acima segue um formato similar ao do relatório do CPLEX.

Ao final da execução, seu código deve exibir as seguintes informações:

- Melhor solução inteira viável encontrada. Se a busca tiver sido completa, esta deve ser uma solução ótima.
- Número total de nós da árvore de branch-and-bound cuja relaxação linear resolvida.
- Valor do *gap* de integralidade ao final da execução. Se a busca tiver sido completa, este *gap* deve ser igual a zero.
- Opcionalmente, o tempo total decorrido durante a execução do algoritmo de B&B.

[1] Que ordena os itens em ordem decrescente da razão $\frac{v_i}{p_i}$ e inclui a maior quantidade possível de cada item, até esgotar a capacidade da mochila.

Nos quadros acima, falamos do caso em que a busca é completa. Isso diz respeito a um limite que seu código deve ter em relação ao número de nós avaliados (isto é, cuja relaxação linear foi resolvida). Este limite deve ser uma variável ou constante em seu código, **obrigatoriamente chamada de** `MaxNos`. **Se** `MaxNos` **tiver valor zero ou negativo, não há limite sobre o número de nós avaliados. Caso contrário, a execução do algoritmo deve ser interrompida após** `MaxNos` **terem sido avaliados.** Em qualquer um dos casos, o relatório de execução e o relatório final devem ser impressos, conforme descritos nos quadros acima.

É inevitável que, em seu código, você precise fazer escolhas que incluem:

(a) a ordem em que variáveis serão ramificadas;

(b) a ordem de avaliação dos nós ainda não avaliados;

(c) a estrutura de dados que irá armazenar os nós ainda não avaliados.

Todas essas escolhas deve estar documentadas em um documento simples, **em formato PDF**. Se desejar usar este documento para explicar certos detalhes sobre o código que você julga necessários, fique à vontade.

2 Formato dos dados de entrada

A instância a ser resolvida pelo seu algoritmo deve ser lida a partir de um arquivo-texto chamado "`entrada.txt`". O arquivo terá apenas quatro linhas, organizadas assim:

- A primeira linha deve conter um inteiro com o número de itens disponíveis (vamos chamar esse número de n).
- A segunda linha deve conter um inteiro com a capacidade da mochila.
- A terceira linha deve conter n inteiros, separados entre si por um espaço em branco, informando o valor de cada item.
- A quarta linha deve conter n inteiros, separados entre si por um espaço em branco, informando o peso de cada item.

Abaixo, é fornecido um exemplo de arquivo de entrada, com a instância utilizada em uma das aulas sobre o algoritmo de *branch-and-bound*:

```
3
10
45 48 35
5 8 3
```

Algumas instâncias serão disponibilizadas no Solar, mas fique à vontade para gerar suas próprias. Você pode usar, por exemplo, o software Excel para gerar estes dados rapidamente, usando a função "`ALEATÓRIOENTRE(inferior, superior)`".

3 Pontuação

Cinco pontos na nota da segunda unidade. Haverá uma prova para complementar a pontuação.

4 Regras

1. Os itens que devem ser entregues nesta atividade são: um código-fonte (em um ou mais arquivos C++) e um documento PDF. Por favor, **não inclua** arquivos de projeto C++, sejam eles do CodeBlocks, Visual Studio, etc.

2. O nome do arquivo lido pelo seu programa deve estar fixo no seu código e deve ser idêntico a "`entrada.txt`". O nome não deve ser solicitado ao usuário.

3. Recomenda-se que esta atividade seja feita em dupla. Em caso excepcional, uma equipe de três participantes poderá ser constituída, com a aprovação prévia do professor.

4. Não copiem código de outras fontes. Você podem ficar à vontade para consultar códigos existentes, ou trocar ideias com colegas, mas não copiem código (mesmo que pretendam fazer mudanças posteriores). Façam suas próprias escolhas e estejam prontos(as) para explicá-las para mim após a entrega da atividade, em uma sessão síncrona via Google Meet.

5. Fiquem à vontade para me consultar a qualquer tempo, via e-mail ou no fórum correspondente do Solar.

6. O prazo de entrega é dia 16 de outubro de 2020, 23:59:59, via e-mail. É possível enviar antes do prazo.

7. Releia com calma a descrição neste documento, para não esquecer algum detalhe e perder pontuação. As informações essenciais estão nos quadros, ou marcadas em negrito no texto.

5 Trilha Sonora Sugerida

"Cantando num fôlego só", de Francis Bringell (`https://youtu.be/bbOwpqyByTM`).