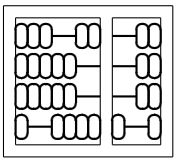

Universidade Estadual de Campinas - UNICAMP
Instituto de Computação - IC

MC658 – Projeto e Análise de Algoritmos III
Prof. Cid Carvalho de Souza
Trabalho Prático – 2º semestre de 2017

1 Problema

Na produção de filmes, as diferentes cenas são geralmente filmadas em uma ordem diferente da que é apresentada na edição final. Diversos aspectos são levados em consideração ao se escolher a ordem de gravação. Naturalmente, são raras as cenas em que todo o elenco deve participar simultaneamente. Por isso, após definida a ordem de gravação das cenas, cria-se uma escala, indicando, para cada ator, os dias que ele deve estar presente para as filmagens. Especialmente em produções distantes do local de residência do elenco, não é viável que cada pessoa viaje de volta para casa sempre que não está escalada. Os dias nos quais um ator está presente na localização das filmagens mas não é requisitado são conhecidos como *dias de espera*.

Cada membro do elenco tem um salário fixo por dia, que deve ser pago enquanto o ator estiver disponível no local de filmagem. Portanto, os produtores devem pagar não apenas pelos dias em que o ator é requisitado, mas também por seus dias de espera. Note que é possível reduzir os dias de espera de um ator retardando sua chegada ou antecipando sua partida do local de filmagem. Claramente, o dia de chegada e de partida de um ator estão associados ao primeiro e ao último dia em que ele foi escalado. Logo, a ordem de gravação das cenas influencia o número de dias de espera de cada ator. Ao se planejar o cronograma de gravação, deseja-se minimizar o custo total com dias de espera de todos os atores. Este problema é conhecido como o *Problema do Planejamento de Cronograma com Custo de Espera Mínimo*.

Seja m o número de atores e n o número de dias de gravação de um determinado filme. Para simplificar a descrição do problema, suponha que a cada dia seja gravada exatamente uma cena. Para indicar em que cenas cada ator é requisitado, definimos uma matriz T de dimensão $m \times n$. Dado um ator i ($1 \leq i \leq m$) e uma cena j ($1 \leq j \leq n$), o elemento t_{ij} de T (i.e., o elemento na linha i, coluna j) tem valor 1 se o ator i está presente na cena j, e 0 caso contrário. Veja um exemplo na Tabela 1. Note que se as cenas fossem gravadas na ordem exibida no filme, o ator 1 teria 3 dias de espera, enquanto que o ator 8 não teria nenhum. Em geral, não é possível eliminar os dias de espera de todo o elenco. Por isso, a melhor ordenação das cenas depende do custo associado com cada ator.

Ator	Cenas																											
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	1	1	1	1	0	1	1	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	0	0	0	0	1	1	0	0	1	1	1	0	0	1	1	1	1	1	0	1	0	0	0	0	1
3	0	1	1	0	1	0	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	0
7	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
8	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabela 1: Matriz T para um filme com $m = 8$ atores e $n = 28$ cenas.

Para armazenar os salários dos atores, definimos um vetor de custos C, de m elementos. Para cada ator i, o i-ésimo elemento de C é denotado por c_i e corresponde ao seu salário diário. Para a instância da Tabela 1, temos $C = [1000, 400, 500, 500, 500, 4000, 400, 2000]$. Seja h_i o número de dias de espera do ator i em uma dada ordem de filmagem das cenas. O problema em questão consiste em encontrar uma ordem de gravação que minimiza $\sum_{i=1}^{m} c_i \cdot h_i$. É conhecido que este problema é NP-difícil.

2 Métodos de Resolução

Neste trabalho, você deverá implementar três métodos diferentes de resolução para o problema proposto: um baseado em Programação Linear Inteira (PLI), um *branch and bound*, e uma heurística baseada em uma metaheurística. Os dois primeiros (PLI e *branch and bound*) deverão ser métodos exatos. Os algoritmos deverão ser avaliados experimentalmente utilizando as instâncias providas (veja mais informações na Seção 4). A seguir, são fornecidos mais detalhes sobre cada um dos métodos.

2.1 Heurística

Projete e implemente uma heurística eficiente que encontre soluções viáveis para o problema proposto. Cada grupo está livre para criar a heurística da forma que lhe parecer mais eficaz. No entanto, a heurística deve **obrigatoriamente** basear-se em uma metaheurística.

2.2 Programação Linear Inteira

Para a resolução com PLI, escreva, inicialmente, uma formulação que modele corretamente o problema. Em seguida, implemente sua formulação na linguagem `mathprog`. Finalmente, execute os testes com as instâncias fornecidas, utilizando o GLPK como resolvedor.

2.3 Branch and Bound

Para a implementação do *branch and bound*, é necessário definir ao menos *i*) uma estratégia de construção de soluções; *ii*) um método de cálculo de limitantes inferiores; e *iii*) uma estratégia de exploração da árvore de enumeração. A seguir, descrevemos uma possível abordagem para cada um destes itens. Os grupos deverão implementar pelo menos os métodos descritos aqui. Extensões podem ser adicionadas livremente e algumas ideias são dadas ao final desta seção.

2.3.1 Construção de Soluções

Uma solução para o problema em questão consiste em uma permutação das cenas. Porém, conforme mostramos na seção seguinte, o cálculo de limitantes inferiores é beneficiado se a construção das soluções se der "de fora para dentro". Mais precisamente, definimos primeiro a cena que será gravada no dia 1, em seguida a cena do dia n, depois do dia 2, $n-1$, 3, $n-2$, e assim por diante.

Isso significa que a árvore de *branch and bound* terá a seguinte estrutura. O nó 0 (raiz), no nível 0 da árvore, representa uma solução vazia, na qual ainda nenhuma cena foi programada. Os nós no nível 1 da árvore representam as soluções parciais nas quais já foi definida a cena do dia 1. Como há n cenas, o nó raiz tem exatamente n filhos. Os nós no nível 2 da árvore representam soluções parciais nas quais já foi definida a cena a ser gravada nos dias 1 e n. Cada nó do nível 1 tem exatamente $n-1$ filhos. Logo, há $n(n-1)$ nós no nível 2 da árvore, e assim por diante. No total, a árvore terá $n+1$ níveis. O último será o nível n, no qual haverá exatamente $n!$ nós, cada um dos quais representa uma das permutações da lista de cenas.

2.3.2 Limitantes Inferiores

O cálculo de limitantes inferiores é fundamental para evitar que todos os nós da árvore sejam explorados explicitamente. No texto que segue, descrevemos como calcular um limitante inferior simples, e depois mostramos como fortalecê-lo.

Seja P uma solução parcial. De acordo com nosso método de construção, P já tem algumas cenas programadas para os primeiros dias de filmagem e também para os últimos. Os dias intermediários ainda estão indefinidos. Um exemplo é mostrado na Tabela 2 utilizando os mesmos dados de entrada da Tabela 1. Seja $E(P)$ o conjunto de cenas já programadas à esquerda em P (i.e., nos dias iniciais). Analogamente, seja $D(P)$ o conjunto de cenas já programadas à direita em P (i.e., nos dias finais). No exemplo dado, $E(P) = \{2, 13, 28\}$ e $D(P) = \{7, 10, 22\}$.

Inicialmente, note que se o primeiro e o último dia de filmagem de um ator já foram determinados em P, é possível calcular exatamente quantos dias de espera ele terá. Isso ocorre em dois casos: *i*) se um ator está escalado para pelo menos uma das cenas de $E(P)$ e também de $D(P)$; ou *ii*) se todas as cenas de um ator já foram programadas em $E(P)$ ou em $D(P)$. Em nosso exemplo, o caso *i* ocorreu com o ator 1. Consultando a Tabela 1, vemos que ele participa de apenas 12 cenas no total. Porém, em P, ele estará

Ator	Cenas						
	28	02	13	...	07	22	10
1	0	1	1		1	0	1
2	1	1	1		0	0	0
3	0	1	0		0	0	0
4	0	0	1	...	0	0	0
5	1	1	0		0	0	0
6	0	0	0		0	1	0
7	0	0	0		0	0	0
8	0	0	0		1	0	1

Tabela 2: Solução parcial P. As cenas 28, 2 e 13 foram programadas para os três primeiros dias, enquanto que 7, 22 e 10 foram programadas para os três últimos, respectivamente.

presente no local de filmagem desde o segundo dia de gravação até o último. Logo, ele deverá ser pago por 27 dias, resultando em 15 dias de espera.

De forma geral, seja $A(P)$ o conjunto de atores cujos dias inicial e final de filmagem já são conhecidos em P. Dado um ator $i \in A(P)$, sejam $e_i(P)$ e $d_i(P)$ o primeiro e o último dia nos quais i está escalado em P, respectivamente. O número de dias que i permanece no local das gravações é $d_i(P) - e_i(P) + 1$. Denotamos também por s_i o número de cenas das quais o ator i participa (i.e., $s_i = \sum_{j=1}^{n} t_{ij}$). Um limitante inferior $K_1(P)$ para o custo da solução parcial P é dado por

$$K_1(P) = \sum_{i \in A(P)} c_i \cdot (d_i(P) - e_i(P) + 1 - s_i) \ . \tag{1}$$

O limitante derivado acima, embora seja válido, é fraco. Isso se deve ao fato de que consideramos apenas os atores cujo número de dias de espera já está totalmente determinado em P. Em nosso exemplo, isso aconteceu para um único ator. Um limitante melhor pode ser obtido se, além dos cálculos já descritos, analisarmos também os demais atores. Seja $B_E(P)$ o conjunto de atores que não estão em $A(P)$, estão escalados para pelo menos uma cena de $E(P)$ mas não estão em nenhuma de $D(P)$. Analogamente, seja $B_D(P)$ o conjunto de atores que não estão em $A(P)$, estão escalados para pelo menos uma cena de $D(P)$ mas não estão em nenhuma de $E(P)$. Em nosso exemplo temos $B_E(P) = \{2, 3, 4, 5\}$ e $B_D(P) = \{6, 8\}$. Se um ator i está em $B_E(P)$, apenas $e_i(P)$ é conhecido, enquanto que se ele está em $B_D(P)$, apenas $d_i(P)$ é conhecido.

Note que, em nosso exemplo, o terceiro dia será, obrigatoriamente, um dia de espera para os atores 3 e 5. Da mesma forma, os dias $n-1$ e $n-2$ serão dias de espera para os atores 8 e 6, respectivamente. Sejam $l_E(P)$ e $l_D(P)$ $(1 \leq l_E(P) < l_D(P) \leq n)$ o último dos dias iniciais e o primeiro dos dias finais cujas cenas já estão determinadas em P, respectivamente. Em nosso exemplo, temos $l_E(P) = 3$ e $l_D(P) = n - 2$. Para um dia l já programado em P $(1 \leq l \leq l_E(P)$ ou $l_D(P) \leq l \leq n)$, denotamos por $P(l)$ a cena a ser filmada no dia l. De forma geral, para um ator i em $B_E(P)$, todos os zeros que

aparecem na Tabela 2 na linha i entre $e_i(P)$ e $l_E(P)$ correspondem a dias de espera. Analogamente, para um ator i em $B_D(P)$, todos os zeros que aparecem na Tabela 2 na linha i entre $l_D(P)$ e $d_i(P)$ também são dias de espera. Assim, um novo limitante inferior $K_2(P)$ é dado por

$$K_2(P) = \sum_{i \in B_E(P)} c_i \cdot \left(\sum_{l=e_i(P)}^{l_E(P)} 1 - t_{iP(l)} \right) + \sum_{i \in B_D(P)} c_i \cdot \left(\sum_{l=l_D(P)}^{d_i(P)} 1 - t_{iP(l)} \right) \quad . \qquad (2)$$

Como $B_E(P)$ e $B_D(P)$ não possuem nenhum dos atores de $A(P)$, $K_1(P)$ e $K_2(P)$ podem ser somados, gerando um limitante inferior mais forte.

Para concluir, é possível também identificar dias de espera para os atores em $B_E(P)$ e $B_D(P)$ entre os dias $l_E(P)$ e $l_D(P)$. Considere, inicialmente, apenas os atores em $B_E(P)$. Seja Q um conjunto de cenas que ainda não foram programadas em P de tal forma que cada ator em $B_E(P)$ tenha no máximo uma cena em Q. O conjunto Q é chamado de um *empacotamento* em relação aos elementos de $B_E(P)$. Em nosso exemplo, as cenas 5, 16 e 23 são um empacotamento em relação a $B_E(P) = \{2, 3, 4, 5\}$, conforme mostrado na Tabela 3.

Ator	Cenas					
	28	02	13	16	05	23
1	0	1	1	0	0	0
2	1	1	1	0	0	1
3	0	1	0	0	1	0
4	0	0	1	1	0	0
5	1	1	0	1	0	0
6	0	0	0	0	0	1
7	0	0	0	0	1	0
8	0	0	0	0	0	0

Tabela 3: Empacotamento Q em relação a $B_E(P) = \{2, 3, 4, 5\}$.

Note que, se as cenas de Q estiverem na ordem mostrada na Tabela 3, os dias 4 e 5 serão de espera para o ator 2 e o dia 4 será de espera para o ator 3. Estes dias de espera sempre ocorrerão para estes atores, independente da posição de outras cenas ainda não inseridas no cronograma (os dias podem ser postergados, mas ainda assim serão de espera).

Naturalmente, as cenas de Q podem aparecer em uma ordem diferente da mostrada na Tabela 3. Por exemplo, se as cenas 16 e 23 trocassem de posição, os dias 4 e 5 seriam de espera para os atores 4 e 5. No entanto, note que a cena de Q filmada por último sempre resultará em pelo menos $|Q| - 1$ dias de espera para os atores de $B_E(P)$ que participam da mesma. Já a cena de Q filmada em penúltimo lugar resultará em pelo menos $|Q| - 2$ dias de espera para os atores envolvidos, e assim por diante.

Uma vez que queremos encontrar limitantes inferiores no custo de uma solução, devemos calcular qual ordem das cenas de Q leva ao custo mínimo de dias de espera,

considerando-se apenas os atores em $B_E(P)$. Este problema pode ser resolvido com um algoritmo guloso, conforme descrevemos a seguir. Note que, em nosso exemplo, a cena 16 contém os atores 4 e 5 de $B_E(P)$, cujos salários são de 500 cada. Logo, cada dia de espera imposto a esses atores tem custo de 1000. A cena 5, por sua vez, tem apenas o ator 3 de $B_E(P)$, cujo salário diário é 500. Finalmente, a cena 23 tem apenas o ator 2 de $B_E(P)$, cujo salário é 400. Assim, por ter o custo menor, a cena 23 deve ser filmada por último. O ator 2 fica, portanto, com 2 dias adicionais de espera, porém ele é o de menor salário. A cena 5 deve ser filmada em segundo lugar, pois tem custo de 500 por dia de espera. A cena 16, por se a mais cara, deve ser filmada o mais cedo possível.

O algoritmo para computar esta última adição aos limitantes é dado a seguir:

1. Compute o conjunto $B_E(P)$.
2. Encontre um empacotamento Q em relação a $B_E(P)$.
3. Para cada cena $j \in Q$, calcule o custo $\tilde{c}_j = \sum_{i \in B_E(P)} c_i \cdot t_{ij}$.
4. Ordene as cenas de $j \in Q$ em ordem decrescente de $\tilde{c}_j$. Seja $Q[j]$ a j-ésima cena da lista Q ordenada.
5. Calcule o limitante $K_3(P) = \sum_{j=0}^{|Q|-1} j \cdot \tilde{c}_{Q[j]}$.

Um procedimento análogo pode ser executado para o conjunto $B_D(P)$, levando a um limitante $K_4(P)$. Os valores obtidos dos quatro limitantes são independentes e podem ser somados, de forma que o limitante inferior final calculado é $K(P) = K_1(P) + K_2(P) + K_3(P) + K_4(P)$.

Os grupos estão livres para projetar um algoritmo para encontrar o empacotamento do passo 2. Sugerimos que seja utilizado um algoritmo guloso rápido que procure maximizar o tamanho do empacotamento.

2.3.3 Exploração da Árvore

Um dos fatores que determinam a eficácia de um *branch and bound* é a estratégia de exploração da árvore. Aqui, sugerimos que, ao menos no início da exploração, seja utilizado um método que rapidamente produza um limitante superior (i.e., uma solução completa viável). Portanto, inicie a busca aplicando a seguinte regra: enquanto não houver uma solução completa, prossiga a exploração da árvore pelo nó ativo de menor limitante inferior dentre todos os nós mais profundos da árvore.

Um exemplo de como aplicar essa estratégia está mostrado na Figura 1 para uma instância hipotética de $n = 4$ cenas. Inicialmente, o nó raiz tem limitante inferior 0. A busca começa pela raiz, a qual dá origem a quatro filhos. Os nós mais profundos da árvore são, nesse instante, os que se encontram no nível 1. O de menor limitante inferior (6) é, então, explorado, dando origem a outros três filhos no nível 2. Este é, agora, o nível mais profundo, e a busca segue respeitando a mesma regra. No último nível, há apenas um nó e o valor indicado corresponde ao valor da solução completa

obtida. Note que, como esse valor é 14, já é possível podar um dos nós do nível superior por limitante. Assim, o nó com limitante inferior 15 é removido da lista de nós ativos. Se continuarmos aplicando a mesma estratégia, o próximo nó a ser explorado é o de limitante inferior 10, no nível 2 da árvore.

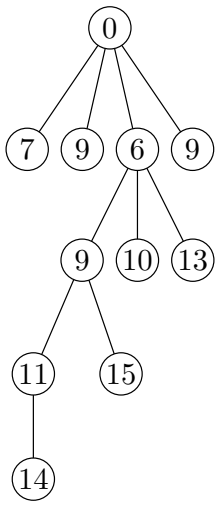

Figura 1: Exploração inicial da árvore de *branch and bound*. Os números indicam o limitante inferior calculado em cada nó.

Os grupos estão livres para decidir como proceder com a exploração após encontrar a primeira solução completa. A mesma estratégia pode ser mantida, ou outra pode ser utilizada, de acordo com o que o grupo considerar mais eficaz.

2.3.4 Extensões

Os métodos descritos até aqui são suficientes para obter-se um algoritmo de *branch and bound* correto. Porém, algumas extensões podem torná-lo mais eficiente. Seguem aqui algumas ideias que os grupos podem utilizar se desejarem.

Limitantes superiores podem ser obtidos mais rapidamente utilizando-se heurísticas. Em alguns casos, pode ser proveitoso executar a heurística desenvolvida na Seção 2.1 antes de iniciar o *branch and bound*, para assim obter um bom limitante superior. Além disso, se o grupo desenvolveu uma busca local, é possível aplicá-la a cada solução completa encontrada na busca (note que soluções completas são obtidas sempre que a busca chega ao último nível da árvore). É possível, ainda, aplicar heurísticas a soluções parciais.

3 Implementação

As heurísticas e o algoritmo de *branch and bound* deverão necessariamente ser desenvolvidos em `C` ou `C++` e compiláveis em ambiente Linux usando o compilador `gcc` ou

`g++`. Já o modelo PLI deverá ser desenvolvido usando o `Mathprog` para ser resolvido pelo `GLPK`, como nos exemplos mostrados em aula, e também deverá rodar em ambiente Linux.

O arquivo `pli.mod` contendo o modelo PLI será entregue parcialmente preenchido, devendo apenas ser completado. Os programas contendo a heurística e o algoritmo *branch and bound* deverão chamar-se respectivamente `heur.*` e `bnb.*`, onde `*` é a extensão apropriada que depende da linguagem de programação utilizada.

Detalhes completos sobre a implementação serão divulgados até o dia 10 de novembro.

4 Experimentos

As instâncias de entrada para as heurísticas e para o algoritmo de terão os formatos de entrada e saída especificados a seguir. Haverá um tempo limite de execução para todos os programas.

Detalhes completos sobre os experimentos serão divulgados até o dia 10 de novembro.

Formato de entrada. Na Figura 2 vê-se um arquivo de entrada para uma instância com 8 cenas e 6 atores no formato `txt`. A primeira linha contém o número de cenas (n), a segunda linha o número de atores (m), as m linhas seguintes referem-se à matriz binária T com dimensão $m \times n$ (onde $T[i][j] = 1$ se e somente se o ator i participa da cena j). Finalmente, a última linha contém m valores inteiros positivos, sendo o i-ésimo valor correspondente ao custo diário do ator i. Ao lado encontra-se o arquivo representando a mesma instância no formato "`.dat`" que será lida pelo modelo PLI implementado no `Mathprog`. Note que o código parcialmente implementado no arquivo `pli.mod` disponibilizado já está preparado para ler estes dados. Mas, para isso, é fundamental que o nome dos parâmetros usados **não sejam mudados!**

Formato de saída. O formato de saída dos programas é o seguinte:

1. Linha 1: n valores inteiros separados por pelo menos um espaço em branco, onde o j-ésimo valor inteiro corresponde ao dia (posição na sequência) em que foi gravada a cena j na melhor solução encontrada pelo algoritmo;

2. Linha 2: um valor inteiro correspondente ao custo (total) da melhor solução encontrada pelo algoritmo;

3. Linha 3: no caso da heurística e do modelo PLI, **esta linha <u>não</u> deve ser impressa**. Já no caso do algoritmo *branch and bound* deve ser impresso um valor inteiro correspondente ao teto do melhor limitante dual encontrado, o qual deverá ser igual ao valor da linha anterior sempre que a otimalidade da melhor solução encontrada tiver sido provada pelo algoritmo.

```
8
6
0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0
0 1 0 1 1 1 1 1
0 1 0 1 1 1 0 0
0 1 1 1 0 0 1 0
0 1 1 1 0 0 0 0
5 4 20 5 5 4
```

```
data;
param n := 8;
param m := 6;
param T : 1 2 3 4 5 6 7 8 :=
 1 0 0 0 0 0 0 0 1
 2 1 0 0 0 0 0 0 0
 3 0 1 0 1 1 1 1 1
 4 0 1 0 1 1 1 0 0
 5 0 1 1 1 0 0 1 0
 6 0 1 1 1 0 0 0 0;
param c :=
 1 5
 2 4
 3 20
 4 5
 5 5
 6 4;
end;
```

Figura 2: Arquivo de entrada no formato `txt` e `dat` (mesma instância).

No caso do exemplo acima, a saída para o modelo PLI (solução ótima) conterá apenas as seguintes linhas:

```
1 4 2 5 6 7 3 8
4
```

5 Relatório

Os grupos deverão entregar um relatório com **no máximo 15 (quinze) páginas** contendo pelo menos os itens listados a seguir.

- Descrição, em alto nível, da heurística proposta.
- Descrição do modelo de PLI proposto explicando, sucintamente, qual o significado de cada variável e restrição. Deve ser feita uma análise também do número de variáveis e restrições no modelo em função de m e n (use notação $O(..)$).
- Descrição de qual foi a estratégia de exploração da árvore adotada no *branch and bound*. Não é necessário reproduzir a descrição do método de geração de limitantes inferiores da Seção 2.3. Forneça apenas uma descrição **sucinta** de quaisquer mudanças ou extensões que tenham sido feitas no método proposto aqui (se houver alguma).

- Relato dos resultados experimentais obtidos com as instâncias fornecidas. Pelo menos os seguintes dados devem ser reportados para cada um dos métodos: *i*) valor da melhor solução encontrada; e *ii*) tempo total de execução. Para o algoritmo baseado em PLI e para o *branch and bound*, reporte também: *iii*) valor do melhor limitante inferior; e *iv*) número de nós explorados.

- Analise sucintamente os resultados obtidos e compare os três métodos.

6 Entrega

Detalhes completos sobre a forma de entrega do trabalho, incluindo os formatos dos arquivos fonte dos programas e do relatório, serão divulgados até o dia 10 de novembro.