

Trabalho de PLI - MC548

Ivan Cunha Santana (016331)
Fábio Stasiak Vendramin (002981)
Campinas, 1 de Julho de 2005

Sumário:

Problema 1 - Equicorte Mínimo

1.1 Descrição das variáveis do modelo	2
1.2 Descrição das restrições	2
1.3 Descrição da função objetivo	2

Problema 2 - Coleta de Lixo

2.1 Descrição das variáveis do modelo	3
2.2 Descrição das restrições	3
2.3 Descrição da função objetivo	3

Problema 3 - Restaurantes

3.1 Descrição das variáveis do modelo	4
3.2 Descrição das restrições	4
3.3 Descrição da função objetivo	4

Problema 4 - Caminhão Tanque

4.1 Descrição das variáveis do modelo	5
4.2 Descrição das restrições.	5
4.3 Descrição da função objetivo	5

Problema 1 - Equicorte Mínimo

1.1 Descrição das variáveis do modelo

vertU: array(1..n) of mpvar (vetor de binários)

Vale 1 se o vértice pertencer a U e 0 caso contrário.

acorte: array(1..n,1..n) of mpvar (vetor de binários)

Vale 1 se a aresta pertencer a U e 0 caso contrário

1.2 Descrição das restrições

Restrição 1:

O Subgrafo U deve conter mais ou menos metade do número de vértices do grafo de entrada, essa soma é igual a $|vertU|$ e portanto deve ser igual a $\text{floor}(n/2)$.

$$\text{VerticesU} := \sum (i \text{ in } 1..n) \text{ vertU}(i) = \text{floor}(n/2)$$

Restrição 2:

Para todas as arestas de G, se a aresta pertence ao corte, um dos vértices dela deve valer 1 e outro 0. Portanto o módulo $|\text{vertU}(i) - \text{vertU}(j)|$ deve ser 1 caso ela pertença ao corte e 0 no caso contrário.

Para isso fizemos 4 funções lineares:

```
forall(i in 1..n, j in 1..n | (Adj(i,j)=1)) do
    p1(i,j) := acorte(i,j) <= vertU(i) + vertU(j)
    p2(i,j) := acorte(i,j) <= 2 - (vertU(i) + vertU(j))
    p3(i,j) := acorte(i,j) >= vertU(i) - vertU(j)
    p4(i,j) := acorte(i,j) >= vertU(j) - vertU(i)
end-do
```

1.3 Descrição da função objetivo

$$\text{Objetivo} := \sum (i \text{ in } 1..n, j \text{ in } 1..n) \text{ acorte}(i,j)$$

Soma de todas as variáveis “acorte” , que indicam que a aresta com vértices i e j está no corte.

O objetivo é ter o mínimo possíveis de arestas no corte, ou seja que a soma seja o mínimo.

Problema 2 - Coleta de Lixo

2.1 Descrição das variáveis do modelo

construido: array(1..d) of mpvar

construido: Vetor de binários com todos os locais, que indica se deve ser construído um determinado depósito. Assim, para cada construido(i) temos um valor 1 ou 0 indicando ou não que o depósito deve ser construído no local.

alocado: array(1..d,1..s) of mpvar

alocado: Vetor de binários com todos os locais, que indica se o depósito “d” está alocado para o setor “s”.

2.2 Descrição das restrições

Restrição 1:

Se um depósito não for construído, nenhum setor pode ser alocado a ele:

```
forall (j in 1..d, i in 1..s)
    rest1(i,j) := construido(j) >= alocao(j,i)
```

Restrição 2:

Apesar de podermos ter vários setores alocando o mesmo depósito, um setor não pode estar alocado a mais de um depósito:

```
forall (i in 1..s)
    rest2(i) := sum (j in 1..d) alocao(j,i) = 1
```

Restrição 3:

A soma das capacidades de produção de lixo dos setores alocados a um determinado depósito não pode ultrapassar sua capacidade:

```
forall (j in 1..d)
    rest3(j) := sum (i in 1..s) alocao(j,i) * demanda(i) <= capacidade(j)
```

2.3 Descrição da função objetivo

Para calcular o nosso custo objetivo temos duas somatórias que serão minimizadas:

Objetivo := (sum (c in 1..d) construido(c) * custo_fixo(c)) + (sum(j in 1..d, i in 1..s) alocado(j,i) * demanda(i) * custo_km * distancia(i,j) * 1000)

Procuramos encontrar o custo mínimo para construir os depósitos e atribuí-los aos os setores. Isto levando em conta os custos de construção e de deslocamento do centro de massa do setor até o depósito (transporte do lixo). Foi necessário multiplicar os custos de transporte por 1000 devido à diferença dimensional entre req(s) e custo_km. O custo é dado por metro cúbico transportado e o requisito é em milhares de metros cúbicos.

Problema 3 – Cadeias de Restaurantes

OBS: Nesse problema duas estratégias foram adotadas, a única diferença entre a estratégia a e a b, é que na estratégia b incluímos uma restrição a mais (a restrição 3).

3.1 Descrição das variáveis do modelo

forall (i in 1..n, j in 1..2) create(instalado(i,j))

A variável instalado é uma matriz que verifica se um restaurante de uma determinada cadeia está instalado em um dos possíveis locais para sua instalação.

instalado(i,j) = 1 os caso um restaurante da cadeia j for instalado no local i.
instalado(i,j) = 0 caso contrário.

3.2 Descrição das restrições

Restrição 1:

Um local não pode ter mais de uma instalação. Para isso ser satisfeito basta impossibilitar que a cadeia de restaurantes '1' E a '2' estejam instaladas no mesmo local.

forall(i in 1..n)
Rest1(i) := instalado(i,1) + instalado(i,2) <= 1

Restrição 2:

Se um restaurante de determinada cadeia é instalada em um determinado local, não podemos ter nenhum outro restaurante da mesma

cadeia há uma distância inferior a 5 km. O problema 2-a já é resolvido com essa restrição e a 1a restrição.

```
forall(i in 1..n, j in 1..n, a in 1..2 | (i<j) )  
Rest2(i,j,a) := instalado(i,a) + instalado(j,a) +  
Menos5(i,j) <= 2
```

Restrição 3 (APENAS PARA A PARTE B DO PROBLEMA:

Se um restaurante é instalado em um local, obrigatoriamente pelo menos teremos que instalar outro restaurante de outra cadeia a menos de 5 km do restaurante instalado.

```
forall(i in 1..n) do  
    Rest3(i) := sum(j in 1..n | Menos5(i,j)=1) instalado(j,1) >= instalado(i,2)  
    Rest4(i) := sum(j in 1..n | Menos5(i,j)=1) instalado(j,2) >= instalado(i,1)  
end-do
```

3.3 Descrição da função objetivo

Objetivo := sum (i in 1..n, j in 1..2) lucro(i,j) * instalado(i,j)

Essa função será maximizada para obtermos o lucro máximo. Ela é a soma do lucro mensal de todos restaurantes instalados.

Problema 4 - Caminhão Tanque

4.1 Descrição das variáveis do modelo

forall (j in 1..n, i in 1..m) create(usado(j,i))

usado: variável binária indica o uso de um tanque. Se for igual à 1 o tanque está sendo usado, 0 caso contrário.

forall (j in 1..n, i in 1..m) create(quantidade(j,i))

quantidade: quantidade de combustível que um determinado tanque possui.

4.2 Descrição das restrições

Restrição 1:

Não podemos armazenar mais do que um tanque suporta. Para isso comparamos a quantidade que está sendo armazenada nele com o quanto ele pode armazenar (caso esteja sendo usado), esta restrição implica também que se ele está armazenando algo, ele tem que estar marcado como usado.

1)

forall(j in 1..n, i in 1..m)

CapacidadeMax(j,i) := quantidade(j,i) <= usado(j,i)*capacidade(j)

Restrição 2:

A soma de todos os armazenamentos de um tipo de combustível não pode ser maior que sua demanda:

forall(i in 1..m)

ArmazenamentoMax(i) := sum(x in 1..n) quantidade(x,i) <= demanda(i)

Restrição 3:

Um tanque só pode conter um tipo de combustível, para isso é só verificar se mais de um tipo de combustível está sendo usado em algum tanque. Essa restrição também obriga que um tanque têm que estar sendo usado por pelo menos um combustível.

```
forall(j in 1..n)  
    RestrigeComb(j) := sum(x in 1..m) usado(j,x) = 1
```

4.3 Descrição da função objetivo

Buscamos a sobra mínima de combustível. Temos então que somar toda a demanda e subtrair da quantidade de combustível alocada no caminhão. Minimizamos o resultado

```
Objetivo := sum (x in 1..m) demanda(x) - sum(j in 1..n, i in 1..m)  
quantidade(j,i)
```

```
minimize(Objetivo)
```

This document was created with Win2PDF available at <http://www.daneprairie.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.