**INSTITUTO DE COMPUTAÇÃO - UNICAMP**
**MC658 - Análise de Algoritmos III**

**Docente: Prof. Cid Carvalho de Souza**
**PED: Natanael Ramos**

# 2º Trabalho Prático

**<u>Grupo 10</u>:**

José Ribeiro - RA : 176665
Rodrigo de Sousa Serafim da Silva - RA : 118607 (Sem contribuição)
Rafael Marques Miorim - RA : 157065 (Sem contribuição)

IC/UNICAMP
2019

# 1 Modelagens dos problemas

## 1.1 [gt54]

1. Definição das variáveis:

$$y_i = \begin{cases} 1, & \text{se o vértice } i \text{ pertence ao caminho.} \\ 0, & \text{caso contrário.} \end{cases}$$

   Existirão $|V|$ variáveis $y$.

$$x_{ij} = \begin{cases} 1, & \text{se a aresta } (i,\, j) \text{ pertence ao caminho.} \\ 0, & \text{caso contrário.} \end{cases}$$

   Existirão $|V| * |V|$ variáveis $x$.

2. Definição das restrições:

   (a) O número de arestas ativas que entram num vértice deve ser igual ao número de arestas que saem dele, para todo vértice diferente das extremidades (vértices s e t):

$$\sum_{j \in V} (x_{ij} - x_{ji}) = 0, \forall i \in V, tal\ que\ i \neq s\ e\ i \neq t$$

   Existirão $|V|$ restrições deste tipo.

   (b) Deve ter uma única aresta ativa de saída no vértice $s$:

$$\sum_{j \in V} x_{sj} = 1$$

   Existirá uma restrição deste tipo.

   (c) Deve ter uma única aresta ativa de entrada no vértice $t$:

$$\sum_{j \in V} x_{jt} = 1$$

   Existirá uma restrição deste tipo.

(d) Se pelo menos uma aresta incide no vértice $i$, então o mesmo deve estar no caminho:

$$\sum_{j \in V}(x_{ij} + x_{ji}) = 2 * y_i, \forall i \in V, i \neq s \ e \ i \neq t$$

$$\sum_{j \in V}(x_{ij} + x_{ji}) = y_i, i = s \ ou \ i = t$$

Existirão $|V|$ restrições deste tipo.

Obs: o valor 2 é utilizado, pois no máximo uma única aresta pode entrar e no máximo uma única aresta pode sair.

(e) Para todo par $(i, j)$ pertencente a $C$, no maximo um único vértice do par pode estar no caminho

$$y_i + y_j \leq 1, \forall (i, j) \in C$$

Existirão $|C|$ restrições deste tipo.

(f) Todas as variaveis devem ser binarias:

$$y_i \in \{0, 1\}, \forall i \in V$$

Existirão $|V|$ restrições deste tipo.

$$x_{ij} \in \{0, 1\}, \forall i \in V, \forall j \in V$$

Existirão $|V| * |V|$ restrições deste tipo.

3. Funcao objetivo:

$$min \ z = \sum_{i \in V} \sum_{j \ inV} w_{ij} * x_{ij}$$

Onde

$$w_{ij} = \begin{cases} e, & \text{se a aresta } (i, j) \text{ de peso } e \text{ pertence a A.} \\ M, & \text{caso contrário.} \end{cases}$$

$$M = soma\ de\ todos\ os\ pesos\ de\ todas\ as\ arestas\ do\ grafo$$

### 1.2 [gt10]

Assumimos aqui que "$a$" é a matrix de adjacências do grafo.

1. Definição das variáveis:

$$x_{ij} = \begin{cases} 1, & \text{se aresta (i, j) pertence ao emparelhamento.} \\ 0, & \text{caso contrário.} \end{cases}$$

   Existirão $|V| * |V|$ variáveis $x$.

$$y_i = \begin{cases} 1, & \text{se nodo i tem aresta incidente pertencente ao acoplamento.} \\ 0, & \text{caso contrário.} \end{cases}$$

   Existirão $|V|$ variáveis $y$.

2. Definição das restrições:

   (a) Garante que no máximo uma aresta incidente no nodo i estará no emparelhamento:

$$y_i = \sum_{j \in V} a_{ij} * x_{ij}, \forall i \in V\ e\ i \neq j$$

   Existirão $|V|$ restrições deste tipo.

   (b) (b) Garante que em uma das extremidades da aresta (i, j) incide uma outra aresta que pertence ao emparelhamento. Caso isso não seja verdade, o conjunto de aresta não será maximal:

$$y_i + y_j >= 1, \forall (i, j) \in E$$

   Existirão $|E|$ restrições deste tipo.

(c) Garante a simetria do problema (grafo simples):

$$x_{ij} = x_{ji}, \forall i \in V, \forall j \in V$$

Existirão $|V| * |V|$ restrições deste tipo.

(d) Todas as variaveis devem ser binarias:

$$y_i \in \{0, 1\}, \forall i \in V$$

Existirão $|V|$ restrições deste tipo.

$$x_{ij} \in 0, 1, \forall i \in V, \forall j \in V$$

Existirão $|V| * |V|$ restrições deste tipo.

3. Funcao objetivo:

$$min\ z = \sum_{(i,j) \in E} x_{ij}$$

## 1.3 [ss2]

1. Definição das variáveis:

$$x_{ij} = \begin{cases} 1, & \text{se a tarefa } i \text{ precede a tarefa } j. \\ 0, & \text{caso contrário.} \end{cases}$$

Existirão $|T| * |T|$ variáveis $x$.

$$s_j = instante\ de\ inicio\ da\ tarefa\ j$$

$$y_i = \begin{cases} 1, & \text{tarefa } i \text{ está atrasada.} \\ 0, & \text{caso contrário.} \end{cases}$$

Existirão $|T|$ variáveis $y$.

2. Definição das restrições:

   (a) A tarefa $i$ deve preceder a tarefa $j$, ou o contrário:

   $$x_{ij} + x_{ji} = 1, \forall (i,j) \in TxT \ e \ i \neq j$$

   Existirão $|T| * |T|$ restrições deste tipo.

   (b) A tarefa $j$ deve começar depois da tarefa $i$, caso $i$ precede $j$. Caso $j$ preceda $i$, a expressão deve ser redundante:

   $$s_i + t_i - x_{ji} * M \leq s_j, \forall (i,j) \in TxT \ e \ i \neq j$$

   $$M = \sum_{i \in T} t_i$$

   Existirão $|T| * |T|$ restrições deste tipo.

   (c) A order de precedência das tarefas estabelecida pelo conjunto $S$ deve ser respeitada:

   $$x_{ij} = 1, \forall (i,j) \in S$$

   Existirão $|S|$ restrições deste tipo.

   (d) Se a tarefa $j$ terminar depois do prazo estipulado, então a mesma estará atrasada:

   $$s_j + t_j \leq d_j + y_j * M, \forall j \in T$$

   $$M = \sum_{i \in T} t_i$$

   Existirão $|T|$ restrições deste tipo.

   (e) Variável $s$ deve ser positiva:

   $$s_i \geq 0, \forall i \in T$$

Existirão $|T|$ restrições deste tipo.

(f) Variáveis $x$ e $y$ devem ser binárias:

$$y_i \in \{0, 1\}, \forall i \in T \tag{1}$$

Existirão $|T|$ restrições deste tipo.

$$x_{ij} \in \{0, 1\}, \forall (i, j) \in TxT \tag{2}$$

Existirão $|T| * |T|$ restrições deste tipo.

3. Funcao objetivo: Queremos minimizar o número de tarefas atrasadas:

$$min\ z = \sum_{i \in T} y_i$$

### 1.4 [mn27]

1. Definição das variáveis:

$$x_{ij} = \begin{cases} 1, & \text{se o nodo } i \text{ recebe a cor } j. \\ 0, & \text{caso contrário.} \end{cases}$$

Existirão $|V| * |V|$ variáveis $x$.

$$c_j = \begin{cases} 1, & \text{pelo menos um nodo recebe a cor } j. \\ 0, & \text{caso contrário.} \end{cases}$$

Existirão $|V|$ variáveis $c$.

2. Definição das restrições:

(a) Cada vertice deve ser colorido com exatamente uma cor:

$$\sum_{j \in C} x_{ij} = 1, \forall i \in V$$

Existirão $|V|$ restrições deste tipo.

(b) Para cada aresta pertencente a E, no maximo um vertice do par pode receber a cor j:

$$x_{uj} + x_{vj} \leq 1, \forall(u, v) \in E \ e \ \forall j \in V$$

Existirão $|E| * |V|$ restrições deste tipo.

(c) A cor $j$ devera ser utilizada caso haja no minimo um vertice colorido com a cor $j$:

$$x_{ij} \leq c_j, \forall i \in V \ e \ j \in V$$

Existirão $|V| * |V|$ restrições deste tipo.

(d) Todas as variaveis devem ser binarias:

$$c_i \in \{0, 1\}, \forall i \in V$$

Existirão $|V|$ restrições deste tipo.

$$x_{ij} \in 0, 1, \forall i \in V, \forall j \in V$$

Existirão $|V| * |V|$ restrições deste tipo.

3. Funcao objetivo: Queremos minizar a quantidade de cores utilizadas:

$$min \ z = \sum_{j \in C} c_j$$

## 1.5 [nd16]

1. Definição das variáveis:

$$x_{ij} = \begin{cases} 1, & \text{se o nodo } i \text{ pertence a partição } j. \\ 0, & \text{caso contrário.} \end{cases}$$

Existirão $2 * |V|$ variáveis $x$.

$$y_{uv} = \begin{cases} 1, & \text{se a aresta (u, v) possui um extremo em } V_1 \text{ e outro em } V_2. \\ 0, & \text{caso contrário.} \end{cases}$$

Existirão $|E|$ variáveis $y$.

2. Definição das restrições:

   (a) Todo nodo $i$ deve pertencer a exatamente uma das partições:

   $$x_{i1} + x_{i2} = 1, \forall i \in V$$

   Existirão $|V|$ restrições deste tipo.

   (b) Uma aresta (u, v) terá extremos em V1 e V2 somente se os dois nodos não estiverem na mesma partição:

   $$x_{uj} + x_{vj} + y_{uv} \leq 2, \forall (u, v) \in E \ e \ \forall j \in \{1, 2\}$$

   Existirão $2 * |E|$ restrições deste tipo.

   (c) Todas as variaveis devem ser binarias:

   $$y_{uv} \in \{0, 1\}, \forall (u, v) \in E$$

   Existirão $|E|$ restrições deste tipo.

   $$x_{ij} \in \{0, 1\}, \forall i \in V, \forall j \in \{0, 2\}$$

   Existirão $2 * |V|$ restrições deste tipo.

3. Funcao objetivo:

$$max \ z = \sum_{(u,v) \in E} w_{uv} * y_{uv}$$

# 2 Resultados do Experimento

## 2.1 Dados da máquina

O experimento foi realizado em uma máquina com 15.5 GiB de RAM e Intel® Core$^{TM}$ i7-7700HQ CPU @ 2.80GHz × 8, rodando Ubuntu 18.04.2 LTS. O programa foi executado com Julia em sua versão 1.1.0. Além disso, para criação das planilhas, utilizou-se gnumeric em sua versão 1.12.35, acoplado consigo o solver LPSolve para resolução das PLI no gnumeric. O motivo de não termos utilizado o GLPK, deve-se ao trabalho que teríamos que realizar a mais, onde tal solver deveria ser baixado.

## 2.2 Resultados das formulação da seção 1, em quatro conjuntos distintos de instâncias

| Exercício | 1 | 2 | 3 | Gnumeric |
|---|---|---|---|---|
| [ss2] | 6 | 17 | D=5<br>P=12 | 2 |
| [gt54] | 31 | 638 | — | 56 |
| [gt10] | 8 | 9 | D = 89.00<br>P = 95.00 | 3 |
| [mn27] | 7 | 13 | 23 | 3 |
| [nd16] | 1906 | 8702 | D = 7111193.52<br>P = 6214277.00 | 59842 |

Table 1: Resultados dos programas em julia das formulações da seção 1

Para os resultados da tabela 1 acima, foi utilizado um tempo limite de 5 minutos. Também, para a instância 3 do problema gt54, o tempo de cômputo do programa em julia foi muito grande, de forma que nem os limites duais e primais foram reportados (por isto o campo encontra-se vazio).