

# Programação Inteira - 2IS

José Joaquim de Andrade Neto

February 2018

## 1 O problema do 2 Conjuntos Independentes de Vértices

Dado um grafo simples  $G(V, A)$ , onde  $V$  é o conjunto de vértices e  $A$  é o conjunto de arestas, define-se o problema de 2 Conjuntos Independentes de Vértices (2MIS) como sendo o problema de determinar dois conjuntos independentes de vértices em  $G$ , tal que a soma desses dois conjuntos seja máxima e os conjuntos sejam distintos. Cada conjunto independente é, na verdade, um subconjunto  $B \subseteq V$  tal que não existe aresta  $(u, v)$  que liga os vértices  $u$  e  $v$ . Em outras palavras,  $B$  é um subconjunto de vértices não adjacentes. Para serem considerados distintos, dois conjuntos independentes não devem possuir o mesmo vértice pertencente ao mesmo tempo nos dois conjuntos. Isso significa que, dado um segundo conjunto independente  $D$ ,  $B \cap D = \emptyset$ .

O restante desse trabalho irá apresentar uma formulação inteira para o 2MIS, na qual uma irá trabalhar sob as arestas de  $G$ , enquanto que a outra trabalha baseada nas cliques maximais de  $G$ . Ainda, para a obtenção de soluções de forma mais eficiente, e que essas soluções sejam primais, também são apresentadas duas heurísticas. Todas essas definições foram implementadas nos experimentos.

### 1.1 Formulação Original

O problema é caracterizado por definir dois conjuntos independentes,  $\alpha$  e  $\beta$ , pertencentes ao grafo  $G(V, A)$ , tais que sejam distintos e máximos. Ele pode ser definido da seguinte forma:

$$\begin{aligned}
& \text{maximize} && (\sum_{i \in \alpha} x_i + \sum_{j \in \beta} y_j) \\
& \text{sujeito a} && x_u + x_v \leq 1 \quad \forall (x_u, x_v) \in A \quad (1) \\
& && y_j + y_i \leq 1 \quad \forall (y_j, y_i) \in A \quad (2) \\
& && x_u + y_u \leq 1 \quad \forall u \in \alpha \wedge u \in \beta \quad (3) \\
& && x_i \wedge y_j \in \{0, 1\} \quad \forall i \in \alpha \wedge \forall j \in \beta \quad (4),
\end{aligned}$$

onde  $x_i = 1$  ou  $y_j = 1$  somente se  $i$  ou  $j$  fizerem parte de seus respectivos conjuntos independentes,  $\alpha$  e  $\beta$  (4). As restrições (1) e (2) garantem que os dois conjuntos são independentes, enquanto que a (3) assegura que os conjuntos são distintos, isto é, dado um vértice  $u \in V$ , então ele estará no máximo em um dos dois conjuntos ( $x_u + y_u \leq 1$ ).

## 1.2 Formulação com Cliques

É possível formular o problema do 2MIS usando cliques. Uma clique é um subconjunto de vértices  $X \subseteq V$  tal que todos os pares de vértices são ligados entre si. Em outras palavras, o grafo induzido por  $X$  é completo. Um conjunto independente de vértice não contém mais de um vértice em cada clique. Portanto, dado uma clique  $C$ , têm-se que  $\sum_{u \in C} x_u \leq 1$  para todas as cliques  $C$  pertencentes a  $G$ . Adaptando essa formulação para o problema do 2MIS:

$$\begin{aligned}
& \text{maximize} && (\sum_{i \in \alpha} x_i + \sum_{j \in \beta} y_j) \\
& \text{sujeito a} && \sum_{i \in C} x_i \leq 1 \quad \forall \text{ cliques } C \in \alpha \quad (1) \\
& && \sum_{j \in C} y_j \leq 1 \quad \forall \text{ cliques } C \in \beta \quad (2) \\
& && x_u + y_u \leq 1 \quad \forall u \in \alpha \wedge u \in \beta \quad (3) \\
& && x_i \wedge y_j \in \{0, 1\} \quad \forall i \in \alpha \wedge \forall j \in \beta \quad (4),
\end{aligned}$$

## 2 Heurísticas

Heurísticas permitem que sejam obtidas soluções primais a partir do arredondamento de variáveis fracionárias de soluções lineares ótimas. O arredondamento dá-se a partir da escolha de um vértice  $v$  com  $x_v$  fracional e atribuindo o valor  $x_v = 1$  para ele e  $x_u = 0$  para todos os seus adjacentes. Esse processo é executado até que  $x_v = \{0, 1\}$  para todos os  $v$  dos dois conjuntos. Duas heurísticas são aplicadas, descritas a seguir:

**RND1** : selecione uma variável que maximiza  $\{x_v : x_v < 1\}$

**RND2** : selecione uma variável que minimiza  $\{x_v + \sum_{(x_v, x_u) \in A} x_u : 0 < x_v < 1\}$

Após as duas heurísticas serem aplicadas, suas soluções são comparadas, e a que maximiza a função objetivo é escolhida.

### 3 Implementação

Todos os códigos implementados possuem um esquema padrão para as suas respectivas execuções. Para compilar, basta inserir o comando *make* no diretório dos códigos-fonte. Após a compilação, basta digitar o modo a ser testado (formulação original, com clique, com ou sem heurísticas, etc) através dos argumentos e o caminho para a instância, a qual já está dentro do diretório. Os argumentos para uma formulação deve receber, em ordem, se usará heurísticas (*h*) ou não (*nh*), e se fará o uso de cortes (*c*) ou não (*nc*). Um exemplo que usa heurística mas não corte é descrito a seguir:

```
./2clq1_formulation_c++ h nc ./instancias gl0_5_1
```

Os algoritmos são divididos em fases, sendo estas (i) de leitura do grafo, (ii) pré-processamento (no caso das cliques) e (iii) execução e resultado, ilustrando os dois conjuntos independentes encontrados. A leitura é realizada na função *readgraph()*, a qual redireciona *STDIN* para o arquivo da instância e lê a lista de adjacência do grafo. O pré-processamento trata de computar as cliques (no caso da CLQ1, na função *clq1()*, e adicionar os parâmetros do problema inteiro (restrições, função objetivo, etc).

A execução acontece na função *runOptimization()*. Por fim, a saída do Gurobi também é codificada nessa mesma função. O *Callback* é implementado na classe *mycallback*. Nessa classe estão implementadas as heurísticas e os cortes utilizado nos experimentos.