Relatório Trabalho 2 - Branch and Bound

Pedro Willian Aguiar e Fernando Gbur dos Santos
Universidade Federal do Paraná – UFPR
Curitiba, Brasil

I. Introdução

O presente relatório tem o objetivo de apresentar a implementação e a modelagem da solução para o problema de separação de grupo minimizando conflitos.

II. Descrição do Problema

Em um certo mundo existe um grupo de *n* super-heróis. Alguns dos super-heróis tem conflito entre si, enquanto outros tem grande afinidade, a ponto de só funcionarem juntos. Como neste mundo apareceram dois vilões, os *n* super-heróis resolveram se dividir em dois grupos. Esta separação tem que ser tal que todos os pares de super-heróis com grande afinidade DEVEM ficar no mesmo grupo, e deve também minimizar os pares com conflito dentro de um mesmo grupo.

Considere que cada super-herói é indexado pelos números de 1 a n. Seja *C* o conjunto de pares (a_i,b_i), com $1 \leq i \leq k = |C|$, tais que a_i e b_i tem conflito, e seja *A* o conjunto de pares (a_i,b_i), com $1 \leq i \leq m = |A|$, tais que a_i e b_i tem grande afinidade.

Dados *n*, *C* e *A*, queremos encontrar uma separação em dois grupos não vazios de tal forma que todo par $(x, y) \in A$, *x* e *y* estão no mesmo grupo, e que minimiza o número de pares $(u, v) \in C$ tais que *u* e *v* estão no mesmo grupo.

III. Função limitante dada

A função limitante dada para o uso do bounding é a seguinte:

$B_{dada}(E) = |(x,y) \in C_E|\ g(x) = g(y)\ | + t_E$

Onde *E* é o conjunto dos super-heróis já escolhidos, C_E é o conjunto dos conflitos que envolvem apenas super-heróis com grupos escolhidos, a função g(x) retorna o grupo de certo super-herói e t_E é o número de triângulos em C $\setminus$ C_E que não compartilham nenhum par de super-heróis.

IV. Implementação

A implementação foi feita em linguagem C. O programa gera um arquivo chamado *separa* que ao ser executado, gera uma saída que contém o número de nós criados, o número mínimo de conflitos presentes no grupo, o grupo onde o primeiro super-herói está presente e o tempo de execução.

O programa aceita as opções *-f* e *-o* A opção *-f* desliga os cortes de viabilidade e a opção -o desliga os cortes de otimalidade.

Primeiramente, as variáveis *n*, *k* e *m* são recebidas. Após, são lidos os *k* conflitos e as *m* afinidades. Para armazenar as afinidades e os conflitos, são criadas duas matrizes de tamanho *n*x*n*, onde as linhas representam cada super-herói e as colunas representam a relação que ele tem com os outros super-heróis. Se for igual 1, a relação existe, se for igual a 0, não existe.

Depois, dependendo da opção iniciada com o programa, são calculados os grupos com a menor quantidade de conflitos possível e com todos os super-heróis com afinidade no mesmo grupo.

É apresentado na tela a quantidade de nós explorados, o grupo do super-herói 1 e o tempo de execução da aplicação

A. Exemplo

Para o teste do programa feito foi utilizado o exemplo passado na descrição do trabalho, com as seguintes entradas:

```
4 2 2
1 3
2 4
1 2
3 4
```

4 é a quantidade *n* de super-heróis, 2 é a quantidade *k* de conflitos e 2 a quantidade *m* de afinidades. As linhas seguintes são as atribuições de conflitos e afinidades respectivamente.

As matrizes de relacionamento são criadas e os cálculos para a minimização são iniciados.

Para a função dada pelos os professores o resultado é:

```
8 chamadas
Mínimo de conflitos: 0
Grupo do primeiro super-herói: 1 2
Tempo de execução: 0.040 milisegundos
```

Para o mesmo exemplo sem o bounding:

```
30 chamadas
Mínimo de conflitos: 0
Grupo do primeiro super-herói: 1 2
Tempo de execução: 0.042 milisegundos
```

V. Conclusão

O objetivo deste relatório foi descrever o problema, modelá-lo e apresentar a implementação da solução em linguagem C.

Com a função limitante de bounding foi possível observar a melhoria de desempenho na solução e a grande quantidade de ramos que são cortados devido a sua aplicação.