

## The Stable Matching Problem

**Aluno:** José Eliton Albuquerque Filho

**Orientador:** Thomas Lewiner

### Introdução

O problema do emparelhamento estável (“Stable Matching Problem”), apresentado por David Gale e L.S.Shapley em janeiro de 1962, consiste em encontrar uma relação estável entre dois conjuntos de elementos dados as preferências de cada elemento. Uma relação é dita estável quando não existe nenhum pareamento  $(A, B)$  no qual a relação seria, para ambos, melhor do que a relação com os elementos que eles estão relacionados. Algoritmos para encontrar soluções para o *Stable Matching Problem* tem aplicações em uma gama de situações reais como encontro de parceiros ideais e distribuição de vagas no vestibular, por exemplo.

### Metodologia

Antes da análise do *Stable Matching Problem* (SMP), foram implementados os Algoritmos de Dijkstra e de Ford-Fulkerson, que serviram de base teórica para a implementação do SMP.

O Algoritmo de Dijkstra, o primeiro estudado, calcula o caminho mínimo entre vértices de um grafo direto cujos pesos são não-negativos. Através do relaxamento de vértices, o algoritmo sucessivamente estima os menores caminhos entre um vértice e seus adjacentes e, com o auxílio de uma fila de baixa prioridade, determina o caminho mínimo.

Já o Algoritmo de Ford-Fulkerson interpreta um grafo  $G$  como um fluxo de rede onde cada aresta apresenta uma capacidade não-negativa. Esse método sucessivamente aumenta o valor do fluxo em  $G$  até que a capacidade de alguma aresta chegue a seu limite, determinando o fluxo máximo em  $G$  e a(s) aresta(s) que estão limitando o aumento do fluxo no grafo.

O Problema do Emparelhamento Estável, objetivo principal deste trabalho, consiste na determinação de uma relação estável que envolve dois conjuntos finitos de iguais quantidades de elementos. No caso dos casamentos, que é o caso mais comum envolvendo SMP, esses dois conjuntos são reconhecidos como homens e mulheres. Cada homem cria um ranking entre as mulheres, formando sua lista de preferência. O mesmo ocorre para as mulheres.

Um casamento  $C$  é considerado estável quando não existe um par  $(H, M)$  tal que  $H$  e  $M$ , que não são parceiros em  $C$ , prefiram um ao outro do que seus respectivos parceiros em  $C$ . Esse casamento sempre existe [1].

O Algoritmo de Gale-Shapley (ou G-S), exibido em 1962, consiste no mais comum e elegante método de solução do SMP. Ele possui a seguinte rotina:

**Enquanto** existir homem solteiro que não se propôs à toda mulher **fazer**:

H se propõe à mulher mais preferida à qual não se propôs ainda

**Se** M é livre então:

M e H tornam casal

**Fim se**

**Se** M prefere  $H_n$  ao corrente parceiro então:

Quebrar o par (H, M)

M,  $H_n$  tornam casal

**Fim se**

**Fim enquanto**

Retornar os casais

O Algoritmo G-S pode ser utilizado de duas formas diferentes: uma na qual o homem se propõe à mulher, e outra na qual a mulher se propõe. No caso da proposta feita por homens, cada um deles se propõe para as mulheres de acordo com sua lista de preferências até que se case. Quando uma mulher solteira recebe a proposta, ela casa imediatamente; as casadas, entretanto, comparam o atual marido e o proponente, podendo aceitar ou rejeitar a proposta, de acordo com a lista de preferências dela. Se o casamento é desfeito, o homem se propõe para a próxima da sua lista, até que todos estejam casados.

À primeira vista talvez não seja fácil observar, mas o Algoritmo G-S sempre retorna casamentos estáveis [1].

Por ser bastante restritivo, devemos considerar variações do SMP para tornar sua aplicação mais prática. As mais conhecidas são *Stable Marriage with Incomplete Lists* (SMI), onde as listas de preferências estão incompletas, podendo gerar mais de uma solução, e *Stable Marriage With Ties* (SMT), onde alguns membros podem ser igualmente ranqueados dentro das listas de preferências.

A possibilidade de empates nas listas de preferências nos dão três versões de estabilidade: super, forte e fraca estabilidades. Para explicá-las, antes definiremos o que é um par bloqueante.

Se, em uma relação M, um homem m casa-se com uma mulher w, usemos a notação  $M(m) = w$  e  $M(w) = m$ . Um par bloqueante para M requer as seguintes condições:

i)  $M(m) \neq w$

ii) m prefere w à  $M(m)$

iii) w prefere m à  $M(w)$

Note que M é instável se nele existir um par bloqueante; caso contrário, é estável.

Uma relação é super estável quando não existir um par bloqueante com as seguintes características:

i)  $M(m) \neq w$

ii) m prefere w à  $M(m)$  ou é indiferente

iii) w prefere m à  $M(w)$  ou é indiferente

De maneira similar, temos para uma relação de forte estabilidade:

- i)  $M(m) \neq w$
- ii)  $m$  prefere  $w$  à  $M(m)$
- iii)  $w$  prefere  $m$  à  $M(w)$  ou é indiferente

E, finalmente, de fraca estabilidade:

- i)  $M(m) \neq w$
- ii)  $m$  prefere  $w$  à  $M(m)$
- iii)  $w$  prefere  $m$  à  $M(w)$

Vemos facilmente que os pareamentos de fraca estabilidade sempre existem e pode ser encontrado em tempo polinomial [2]. Forte e super estabilidades, entretanto, nem sempre existem.

## Resultados

O Algoritmo de Dijkstra foi feito na linguagem C. Nele inserimos a quantidade de vértices e as arestas existentes, e obtemos os menores caminhos no grafo dado. Para verificarmos a funcionalidade do código, testamos no seguinte exemplo:

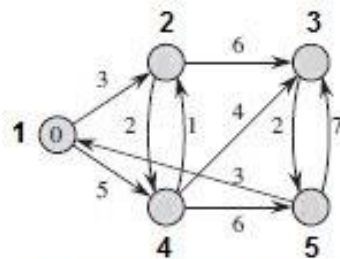


Figura 1: Exemplo de grafo

Os menores caminhos cuja origem é o vértice 1 são:

```

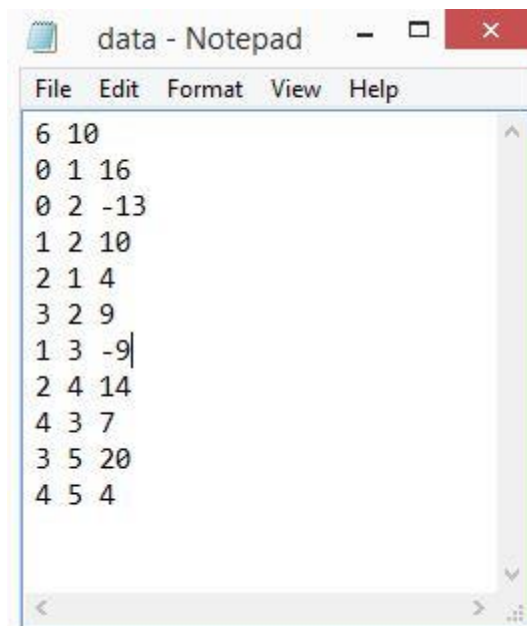
C:\Users\Eliton\Documents\Programs\Dij.exe

Type:
a - Add vertice
r - Run the Algorithm

Lista dos Menores Caminhos no Grafo Dado:
De 1 para 1:      Nao Existe
Custo: -
De 1 para 2:      1 -> 2
Custo: 3
De 1 para 3:      1 -> 2 -> 3
Custo: 9
De 1 para 4:      1 -> 4
Custo: 5
De 1 para 5:      1 -> 4 -> 5
Custo: 11
    
```

Figura 2: Resultado do Algoritmo de Dijkstra

O Algoritmo de Ford-Fulkerson, também desenvolvido em C, funciona um pouco diferente. Ao invés de receber os dados do grafo na própria interface, ele lê um arquivo .txt que contém, da primeira linha, o número de vértices e de arestas, respectivamente, e nas demais linhas, o vértice de origem, o destino e a capacidade dessa aresta.



```
data - Notepad
File Edit Format View Help
6 10
0 1 16
0 2 -13
1 2 10
2 1 4
3 2 9
1 3 -9
2 4 14
4 3 7
3 5 20
4 5 4
```

Figura 3: Dados do Algoritmo de Ford – Fulkerson

O programa retorna o fluxo máximo na rede:



```
C:\Users\Eliton\Documents\Programs\Ford_Fulkerson.exe
10
Process returned 0 (0x0)   execution time : 0.320 s
Press any key to continue.
```

Figura 4: Fluxo máximo da rede

Já no *Stable Matching Problem*, foi implementado um programa em C++ que determina casamentos estáveis. Nesse exemplo utilizamos 11 duplas.

```
const char *men_data[][11] = {
    { "abe", "abi", "eve", "cath", "ivy", "ian", "dee", "fay", "bea", "hope", "gav" },
    { "bob", "cath", "hope", "abi", "dee", "eve", "fay", "bea", "ian", "ivy", "gav" },
    { "col", "hope", "eve", "abi", "dee", "bea", "fay", "ivy", "gav", "cath", "ian" },
    { "dan", "ivy", "fay", "dee", "gav", "hope", "eve", "ian", "bea", "cath", "abi" },
    { "ed", "ian", "dee", "bea", "cath", "fay", "eve", "abi", "ivy", "hope", "gav" },
    { "fred", "bea", "abi", "dee", "gav", "eve", "ivy", "cath", "ian", "hope", "fay" },
    { "gav", "gav", "eve", "ivy", "bea", "cath", "abi", "dee", "hope", "ian", "fay" },
    { "hal", "abi", "eve", "hope", "fay", "ivy", "cath", "ian", "bea", "gav", "dee" },
    { "ian", "hope", "cath", "dee", "gav", "bea", "abi", "fay", "ivy", "ian", "eve" },
    { "jon", "abi", "fay", "ian", "gav", "eve", "bea", "dee", "cath", "ivy", "hope" },
};

const char *women_data[][11] = {
    { "abi", "bob", "fred", "jon", "gav", "ian", "abe", "dan", "ed", "col", "hal" },
    { "bea", "bob", "abe", "col", "fred", "gav", "dan", "ian", "ed", "jon", "hal" },
    { "cath", "fred", "bob", "ed", "gav", "hal", "col", "ian", "abe", "dan", "jon" },
    { "dee", "fred", "jon", "col", "abe", "ian", "hal", "gav", "dan", "bob", "ed" },
    { "eve", "jon", "hal", "fred", "dan", "abe", "gav", "col", "ed", "ian", "bob" },
    { "fay", "bob", "abe", "ed", "ian", "jon", "dan", "fred", "gav", "col", "hal" },
    { "gav", "jon", "gav", "hal", "fred", "bob", "abe", "col", "ed", "dan", "ian" },
    { "hope", "gav", "jon", "bob", "abe", "ian", "dan", "hal", "ed", "col", "fred" },
    { "ivy", "ian", "col", "hal", "gav", "fred", "bob", "abe", "ed", "jon", "dan" },
    { "ian", "ed", "hal", "gav", "abe", "bob", "jon", "col", "ian", "fred", "dan" },
};
```

Figura 5: Listas de preferências

O programa nos dá os passos para determinar os casamentos estáveis, mostrando quando há a quebra de um casamento para a realização de um mais estável.

```
C:\Users\Eliton\Documents\Programs\marriage.exe

Matchmaking:
abi and abe
cath and bob
hope and col
ivy and dan
jan and ed
bea and fred
gav and gav
eve and hal
hope dumped col for ian
abi dumped abe for jon
dee and col
ivy dumped dan for abe
fay and dan
Engagements:
abi and jon
bea and fred
cath and bob
dee and col
eve and hal
fay and dan
gav and gav
hope and ian
ivy and abe
jan and ed
Stability:
<all marriages stable>
```

Figura 6: Determinação dos casamentos estáveis

### **Conclusão**

O Algoritmo G-S é rápido e efetivo mesmo na procura de um grande número de casamentos estáveis. Além disso, o tempo de execução do algoritmo não é muito influenciado pela quantidade de empates no SMT.

As variações do *Stable Marriage Problem* podem ser, na maioria dos casos, resolvidas pelo Algoritmo de Gale-Shapley com as modificações necessárias, tornando possível a aplicação desse problema para casos como distribuição de vagas no Sistema de Seleção Unificada (SISU).

### **Referências**

1 - GALE, D.; SHAPLEY, L.S. **College Admissions and the Stability of Marriage**. The American Mathematical Monthly, Vol. 69, 1962.

2 - R. W. Irving. **Stable marriage and indifference**. Discrete Applied Mathematics, Vol.48, pp. 261–272, 1994.