# Relatório de ASA sobre Controlo de Multidões (Controlo Fluxo)

**Grupo 81:**

André Silva – 68707

Miguel Faria – 73092

## Introdução ao Problema

O segundo projeto tem como motivação a segurança em eventos de elevada afluência (como manifestações, jogos de futebol, etc.) e os algoritmos de controlo de fluxo lecionados em Análise e Síntese de Algoritmos. O problema apresentado descreve vários pontos de concentração de pessoas e os acessos disponíveis a partir desses pontos. É dada atenção especial aos chamados "pontos críticos" – pontos onde a probabilidade de haver problemas de segurança é maior. É proposta a implementação de um programa que, recebendo como entrada: pontos (dos quais vários podem ser pontos críticos) e ligações entre pontos (com indicação do número de problemas entre pontos críticos), identifique, para cada problema, o número mínimo de ligações a barrar, de forma a impedir contacto direto entre um conjunto de pontos críticos, no mapa de uma cidade.

## Descrição da Solução Encontrada

O primeiro passo na abordagem do problema foi perceber de que forma este se enquadrava no âmbito da disciplina. Ao analisarmos a estrutura dos pontos e ligações que representam o mapa da cidade, rapidamente fizemos a analogia com grafos. O barramento de ligações de forma a impedir o alastramento de motins denunciou que o nosso problema se equiparava a um de redes de fluxos.

Na posse desta informação, esboçámos graficamente os *inputs* de exemplo disponíveis no enunciado. Isto permitiu-nos identificar quais as estruturas de dados a usar no projeto para representação desta informação. Decidimos usar, naturalmente, grafos de fluxo. Os grafos de fluxo têm como componentes grafos *standard*, nós, e arcos, organizados hierarquicamente – isto é, a estrutura (classe) dos grafos de fluxo tem um grafo, que tem nós, onde cada nó guarda os arcos que de si partem.

Posteriormente, focámo-nos em como usar esses dados de forma a atingir o nosso objetivo – ou seja, qual seria a abordagem algorítmica ao problema.

Após uma análise cuidada concluímos que a solução do problema era, para cada conjunto de pontos críticos, qual o menor número de caminhos de aumento num grafo de fluxos. De um ponto de vista teórico, isto significa identificar, para cada par de pontos do conjunto de pontos críticos, qual o fluxo máximo que é possível enviar do ponto de origem para o ponto de destino e desses fluxos determinar o mínimo.

Depois de concluirmos, teoricamente, como identificar o número mínimo de caminhos a serem barrados, focámo-nos em identificar algoritmicamente esse número. Determinámos que a melhor forma de o identificar era usar o algoritmo Edmonds-Karp, que faz uma implementação do Método de Ford-Fulkersson usando uma procura em largura no grafo. Desta forma obtém-se, sucessivamente, os caminhos mais curtos entre a origem e o destino. Isto evita um grande problema do Método de Ford-Fulkersson: no caso de um fluxo máximo muito elevado o caminho escolhido entre a origem e o destino pode não ser o mais curto e leva a iterações do algoritmo desnecessárias – e a uma complexidade muito elevada.

De maneira a determinar qual o valor mínimo dos fluxos máximos, executa-se o Edmonds-Karp entre todas as combinações possíveis de valores para origem e destino usando os pontos críticos como origem e destino. Como cada execução do Edmonds-Karp devolve o fluxo máximo num grafo entre um ponto de origem e um de destino, após a execução para um par o valor retornado é comparado com o menor fluxo já encontrado – garantindo assim que apenas o menor valor é guardado. O processo é repetido para todos os pares de pontos críticos e no final é retornado o valor que se encontra na variável de mínimo.

De acordo com as restrições impostas sobre as linguagens nas quais se podia desenvolver o projeto, escolhemos o C++ por estarmos mais familiarizados com esta linguagem e consideramos que permite uma maior abstração dos detalhes da arquitetura da máquina.

## Análise Teórica da Solução

Conforme foi referido na secção anterior, a nossa solução executa, para cada problema, o algoritmo Edmonds-Karp em cada combinação de pontos críticos. Como tal, realiza um ciclo onde em cada iteração são combinados os valores dos pontos críticos e executa-se o Edmonds-Karp com esses pontos como origem e destino.

Visto que o caminho de 1 para 2 tem o mesmo fluxo que 2 para 1 (pois fizemos uso de grafos não dirigidos onde todos os arcos têm peso igual a 1) então o conjunto de pontos é iterado apenas num sentido e cada ponto *i* é combinado com o ponto *j*, começando em *i+1* e terminando no último elemento do conjunto. Tendo este comportamento em consideração conclui-se que a complexidade do ciclo é um $O(n^2)$, onde no pior caso $n = V$, isto é, $n$ é igual ao número de vértices.

Desta forma, o algoritmo usado para determinar o caminho com menos fluxo, para cada conjunto crítico, é um $O(V^3E^2)$, já que a complexidade do algoritmo de Edmonds-Karp é $O(VE^2)$, e como no pior caso é realizado em $O(V^2)$ então tem-se que, nessa situação, o algoritmo tem uma complexidade de $O(V^2)*O(VE^2)$ – o que resulta numa complexidade de $O(V^3E^2)$.

Realizámos esta escolha de algoritmo devido à sua eficiência. Considerando que no ciclo de combinação de pontos de origem e destino não é possível ter complexidade inferior sem se correr o risco de falhar alguma combinação, o algoritmo retorna o fluxo para um par (origem, destino) em tempo $O(VE^2)$. Além da eficiência, a escolha deste algoritmo deveu-se à sua pouca ocupação de memória, pois a maior parte das estruturas que utiliza são pré-existentes. O Edmonds-Karp tem também uma implementação simples, tratando-se de apenas um ciclo onde são realizadas sucessivas procuras em largura, que retornam um valor que é somado ao fluxo já obtido, para obter o fluxo máximo.

## Avaliação Experimental dos Resultados

Aplicámos testes exaustivos ao programa que criámos e concluímos que em casos com 155 pontos, 154 ligações entre esses pontos e 127 problemas onde os conjuntos têm em média 55 pontos e variam entre 10 pontos e 100 pontos, a solução demora 0,02s, o que é um resultado muito positivo considerando o universo descrito.

A tabela apresenta o *output* e o tempo demorado para algumas sequências de dados de entrada.

| Teste Nº | Input | Output Obtido | Output Esperado | Tempo de Execução (s) |
|---|---|---|---|---|
| 1 | 4 4 | 2 | 2 | 0,02 |
| | 0 2 | 2 | 2 | |
| | 0 3 | | | |
| | 1 2 | | | |
| | 1 3 | | | |
| | 2 | | | |
| | 2 0 1 | | | |
| | 2 0 2 | | | |
| 2 | 5 5 | 1 | 1 | 0,01 |
| | 0 2 | | | |
| | 0 3 | | | |
| | 1 2 | | | |
| | 1 3 | | | |
| | 0 4 | | | |
| | 1 | | | |
| | 3 0 4 1 | | | |
| 3 | 48 43 | 0 | 0 | 0,01 |
| | (ligações entre pontos) | 1 | 1 | |
| | 4 | 0 | 0 | |
| | 10 4 13 17 18 23 27 34 37 38 39 | 0 | 0 | |
| | 6 6 12 17 24 27 28 | | | |
| | 7 8 14 17 30 36 39 47 | | | |
| | 6 3 5 13 30 34 38 | | | |
| 4 | 6 4 | 2 | 2 | 0 |
| | 0 4 | | | |
| | 0 5 | | | |
| | 1 4 | | | |
| | 1 5 | | | |
| | 1 | | | |
| | 2 0 1 | | | |

Pela tabela verifica-se que o tempo de execução é muito baixo, demorando menos de 5 centésimas de segundo para todos os testes, o que é bastante positivo já que a solução foi testada tanto com poucos pontos como com muitos, sendo sempre o número de ligações igual ou semelhante ao número de pontos existentes no grafo.

Para finalizar, além dos nossos testes, também submetemos o projeto a uma bateria de 16 testes automáticos realizados pelos professores da disciplina, aos quais passámos a todos dentro do tempo limite.

## Bibliografia


1. *Introduction to Algorithms, Third Edition,* Thomas H. Cormen et al, MIT Press, 2009;
2. *Slides* do corpo docente de ASA disponibilizados na página da disciplina;
3. *Wikipedia.org* – Edmonds–Karp algorithm.