Relatório 2º Projeto ASA 2020/2021

Grupo: al024
Alunos: António Coelho (ist195535) e Gustavo Aguiar (ist195587)

1 Descrição do Problema e da Solução

Pelo enunciado, temos que o pretendido é $min\{\sum_{i \in P_x} \mathrm{X}_i + \sum_{i \in P_y} \mathrm{Y}_i + \sum_{(i,j) \in P_x \times P_y} \mathrm{c}_{ij}\}$. Assim, por definição, este problema trata-se de um corte de capacidade mínima, pelo que com base no Teorema de Fluxo Máximo Corte Mínimo, modelámo-lo como um de rede de fluxo, onde a fonte e o sumidouro correspondem aos processadores X e Y, respetivamente, e os restantes vértices aos n processos do programa. Para além disso, sabendo que $(\sum_{i \in P_x} X_i + \sum_{i \in P_y} Y_i) \in O(n)$, temos que $|f^*| \in O(|V|)$. Justifica-se assim usar um algoritmo de caminhos de aumento, baseado no método de *Ford-Fulkerson*, sendo esses determinados por uma *BFS* - **Algoritmo de *Edmonds-Karp***.

Assim, como auxílio de resolução dos problemas encontrados utilizaram-se as seguintes referências:

- Análise teórica do algoritmo de *Edmonds-Karp*
- Implementação do algoritmo de *Edmonds-Karp* para determinar o fluxo máximo

2 Análise Teórica

A representação da rede residual em memória foi feita com recurso a uma matriz de adjacências - complexidade de espaço $\Theta(V^2)$ - justificada pela bidirecionalidade dos arcos correspondentes aos custos inter-processos e pelo acesso O(1) às capacidades dos arcos na rede residual no decorrer do algoritmo.

A solução visada para a resolução dos problemas apresentados consiste em 2 etapas: ler a rede de fluxo de *input* (1) e computar o fluxo máximo com recurso ao algoritmo de *Edmonds-Karp*(2).

Para (1) - leitura de *input* - usando `scanf`, ler os dados de entrada dentro de dois ciclos (n e k) a dependerem linearmente e quadraticamente de V para os custos entre os processos e os processadores e entre processos, respetivamente. Mais concretamente, $\Theta(V-2) = \mathrm{O(V)}$ para o primeiro ciclo e $\mathrm{O}(\frac{(V-2)(V-3)}{2}) = \mathrm{O(V^2)}$ para o segundo.

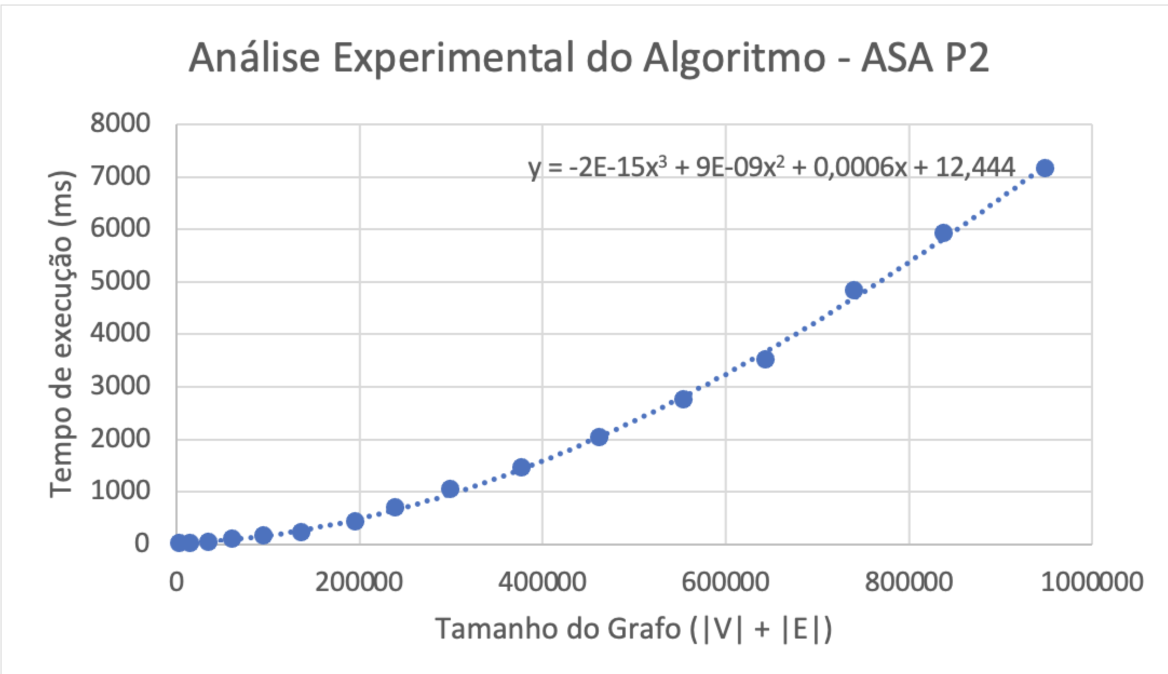
Para (2) - aplicar o algoritmo de *Edmonds-Karp* na rede de fluxo - enquanto houver caminhos de aumento (o que se verifica em $\mathrm{O}(VE)$ vezes) aplica-se a $BFS(\mathrm{O(V^2)}$, uma vez que se trata de uma matriz de adjacências) para encontrar entre esses o mais curto em número de arcos - distância de *Edmonds-Karp* mínima. Existindo um, determina-se a sua capacidade crítica por *backtrack* (que é O(V)) e o fluxo máximo é incrementado esse valor. No total tem-se $\mathrm{O(VE(V^2{+}V)) = O(EV^3)}$.

Por fim, como a complexidade de (2) domina a de (1), tem-se que a solução geral aos problemas apresentados $\in$ $\mathbf{O(EV^3)}$.

3 Avaliação Experimental dos Resultados

Para a devida análise do algoritmo implementado, utilizou-se um computador com processador *Intel Core i5 Quad-Core* a 1.4 GHz, com 8 GB de memória RAM, com o sistema operativo *macOS Big Sur*.

Utilizou-se a ferramenta `gen2procs` fornecida pelo corpo docente para gerar redes de fluxo com capacidade por arco não superior a 15 – mostra-se irrelevante utilizar capacidades muitos superiores uma vez que a solução apresentada tem complexidade independente desse valor - e número de vértices entre 102 e 1602 aumentando de 100 em 100, perfazendo um tamanho de grafo até à ordem de grandeza 10^6. Com o intuito de cronometrar o desempenho do algoritmo implementado utilizou-se a chamada de sistema `time` para o programa a correr sobre as tais redes geradas.



Por fim, registaram-se os valores obtidos da testagem e ajustou-se uma regressão polinomial de ordem 3 que demonstra, experimentalmente, a veracidade da complexidade do algoritmo esperada teoricamente. Como $|E| \gg |V|$, vemos que o tempo de execução está proximamemente relacionado com $|V^3|$, pelo que podemos concluir que o tempo de execução está proximamente relacionado com $\mathbf{O(EV^3)}$.