

3 DE JUNHO DE 2022

# TRABALHO DE DA

AGÊNCIA DE VIAGENS

Henrique Silva up202007242

João Araújo up202004293

Vasco Guedes up202004396

# LOGÍSTICA URBANA PARA VIAGENS TURÍSTICAS

Com a saturação dos serviços de entregas, os acionistas de uma empresa decidiram diversificar o ramo de negócio e apostar também na promoção de viagens turísticas.

A empresa dispõe de veículos em vários locais. Cada um fará um único trajeto de uma origem para um destino, tem uma certa capacidade, e realizará a viagem num certo tempo e com um custo de transporte por pessoa.

Pretende-se um sistema capaz de apoiar a gestão de pedidos para transporte de grupos de pessoas de um local de origem para um local de destino, sendo ambos dados.



# Cenário 1.1

Maximizar a dimensão de um grupo inseparável

## Variáveis relevantes

### Paragens

- origem (**o**,  $\mathbf{o} \in \mathbb{N}$ )
- destino (**d**,  $\mathbf{d} \in \mathbb{N}$ )

### Linhas

- capacidade (**cl**,  $\mathbf{cl} \in \mathbb{N}$ )

## Função objetivo

$$\text{Max } \mathbf{C} = \text{Min}(\mathbf{cl}) \wedge \\ \mathbf{l} \in \{\mathbf{o}, \dots, \mathbf{d}\}$$

Para um conjunto de caminhos compostos por  $\{\mathbf{o}, \dots, \mathbf{d}\}$ , o objetivo seria encontrar o caminho de maior capacidade (**C**,  $\mathbf{C} \in \mathbb{N}$ ), sendo **C** igual à capacidade do ramo ( $\mathbf{l} \in \mathbb{N}$ ) de menor capacidade do caminho.

# Cenário 1.1

## CAMINHOS DE CAPACIDADE MÁXIMA

Utilizou-se o algoritmo dos **Caminhos de Capacidade Máxima** (adaptação do algoritmo de Dijkstra) que permite obter o caminho, entre os pontos **O** e **D**, com maior capacidade.

# Cenário 1.1

## CAMINHOS DE CAPACIDADE MÁXIMA

Tomou-se em consideração  
o problema dos Caminhos  
de Capacidade Máxima com  
adaptação do Algoritmo de  
Dijkstra

Tomando **L** e **P** como 2 vetores dinâmicos que contêm inicialmente, para uma capacidade **V** equivalente ao número de nós, números infinitamente pequenos e 0, respetivamente, **Container** como uma fila de prioridades de pares de valores, onde o valor que se encontra no topo é sempre o maior par, **id** como o número do vértice **V**, **E.dest** como o **id** do nó de destino da aresta e **E.capacity** como a capacidade da aresta.

Inicialmente, coloca-se em **Container** o par (**0**, **O**), equivalente a 0 de capacidade e ao **id** do nó de origem **O**, e inicializa-se **L[O]** com um valor infinitamente grande.

De seguida, enquanto **Container** não estiver vazio, faz-se:

- **tmp** = **Container**.top();
- **current\_O** = **tmp**.second;
- **Container**.pop();
- Para todos os ramos **E** do nó **current\_O**:
  - **capacity** = max(**L[E.dest]**, min(**L[current\_O]**, **E.capacity**));
  - Caso **capacity** seja maior que **L[E.dest]**, faz-se:
    - **L[E.dest]** = **capacity**;
    - **P[E.dest]** = **current\_O**;
    - Coloca-se em **Container** o par (**capacity**, **E.dest**);

# Cenário 1.1

## CAMINHOS DE CAPACIDADE MÁXIMA

Os **id**'s dos nós utilizados no percurso de **O** até **D** estão guardados no vetor dinâmico **P**, e podem ser obtidos pela seguinte função:

Tomando **V** como o **id** do nó corrente e **ret** como um vetor auxiliar que guarda os nós sucessivos.

Enquanto **V** for diferente de **D**, faz:

- $V = P[V]$ ;
- empurra para trás do vetor **ret** o nó **V**;

Os nós utilizados são assim guardados no vetor **ret**. O valor da capacidade do percurso encontra-se armazenada na posição **L[D]**.

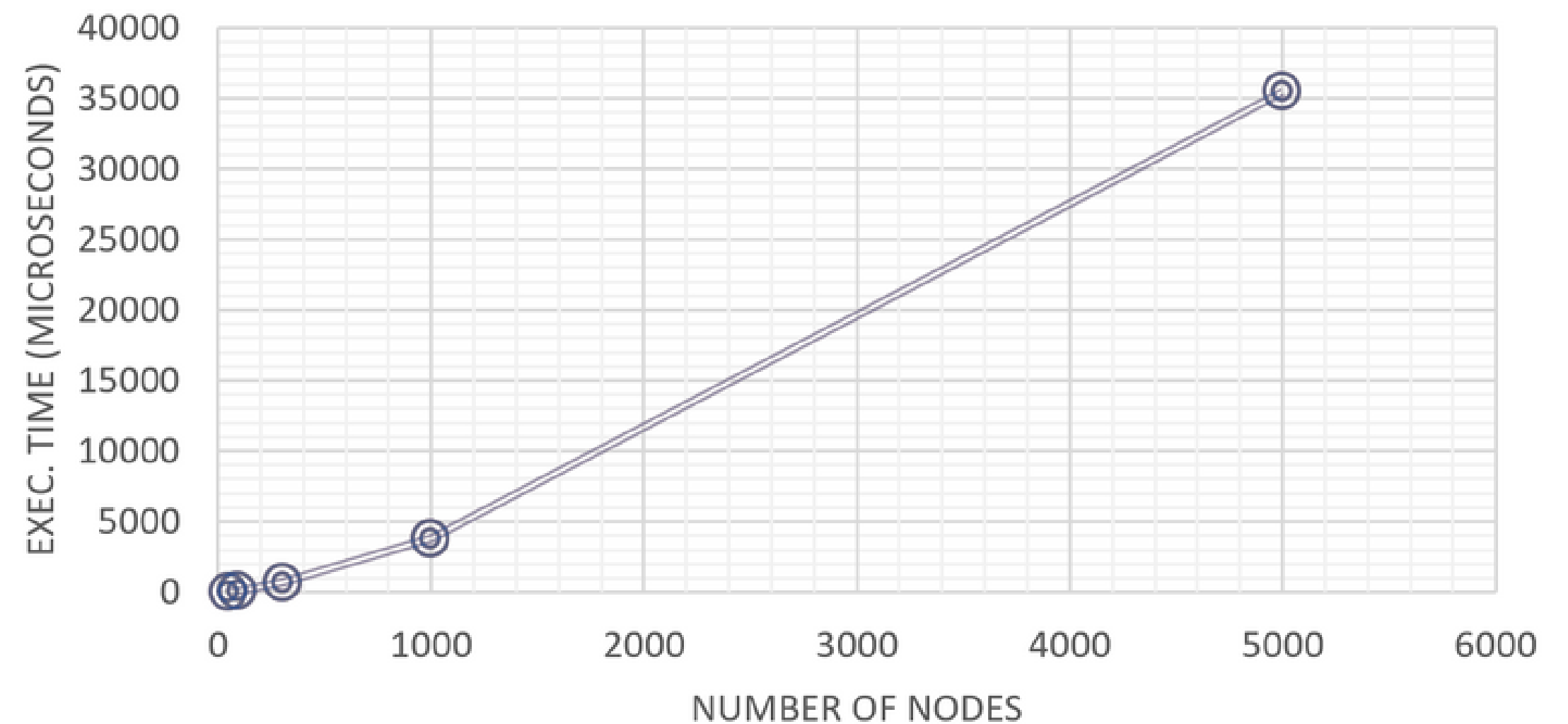
**Complexidade temporal:**  $O((|V| + |E|) * \log_2 |V|)$ , onde **V** corresponde ao número de nós e **E** ao número de arestas. Quanto à capacidade espacial, esta é  $O(|V| + |V| * \log_2 |V|) = O(2 * |V|)$ .

# Cenário 1.1

Avaliação empírica

## Performance 1.1

Vertex	Edges	V * E	Tempo de exec. (μs)
50	136	6 800	35
90	257	23 130	81
300	1417	425 100	679
1000	7533	7 533 000	3836
5000	49487	247 435 000	35469



### Resultados da Avaliação Empírica:

Como referido anteriormente trata-se de uma função linearítmica( $O((|V| + |E|) * \log_2 |V|)$ ).

# Cenário 1.2

Maximizar a dimensão de um grupo inseparável,  
minimizando o número de transbordos

## Variáveis relevantes

### Paragens

- origem (***o***, ***o*** ∈ ℕ)
- destino (***d***, ***d*** ∈ ℕ)

### Linhas

- capacidade  
(***cl***, ***cl*** ∈ ℕ)

## Função objetivo

$$\text{Max } \mathbf{C} = \text{Min}(\mathbf{cl}) \wedge \\ \mathbf{l} \in \{\mathbf{o}, \dots, \mathbf{d}\} \wedge \text{Min } \mathbf{T}$$

Para um conjunto de caminhos compostos por  $\{\mathbf{o}, \dots, \mathbf{d}\}$ , o objetivo seria encontrar os caminhos de maior capacidade (***C***, ***C*** ∈ ℕ), sendo ***C*** igual à capacidade da linha (***l*** ∈ ℕ) de menor capacidade do caminho, assim como de menor número de transbordos (***T***, ***T*** ∈ ℕ). Este problema terá um conjunto de soluções não comparáveis (pareto-ótimas).



## Cenário 1.2

CAMINHOS DE  
CAPACIDADE  
MÁXIMA

+

EDMOND-KARP  
(MÁXIMIZAÇÃO DE FLUXO)

Utilizou-se em conjunção o algoritmo utilizado no cenário anterior com um algoritmo de maximização de fluxo.

O algoritmo dos **Caminhos de Capacidade Máxima** permite obter o caminho, entre os pontos **O** e **D**, com maior capacidade.

A adaptação do algoritmo de Edmond-Karp para a **Maximização de Fluxo** permite obter os restantes caminhos entre os pontos **O** e **D** de tal modo a que o fluxo seja máximo.

# Cenário 1.2

## CAMINHOS DE CAPACIDADE MÁXIMA

ATRÁS EXPLICADO

### EDMOND-KARP (MÁXIMIZAÇÃO DE FLUXO)

Tomou-se em consideração  
o método de Edmond-Karp  
para maximização de fluxo

Tomando **AllPaths** como o vetor dinâmico constituído por múltiplos **Path**'s, que constituí um vetor dinâmico que contém um par de valores que representam, respetivamente, um vetor constituído pelo caminho entre **O** e **D**, **rGraph** como um vetor dinâmico de 2 dimensões que contém um par de valores que representam, respetivamente, o fluxo residual e a capacidade de cada aresta, **V** como o número de nós e **P** como um vetor de capacidade **V**, **E.dest** como o nó de destino de uma determinada aresta, **found** como um booleano, **INT\_MAX** como um valor infinitamente grande.

Inicialmente, vai-se popular o vetor **rGraph**:

- Enquanto **u** < **V** (sendo que **u** inicializa-se a 0):
  - Enquanto **v** < **V**, (sendo que **v** inicializa-se a 0):
    - **found** = falso;
    - Para todos os ramos **E** do nó **u**, faz:
      - Caso **E.dest** seja igual a **v**, faz:
        - **found** = verdadeiro;
        - **rGraph[u][v]** = (E.capacity, E.capacity)
      - Caso **found** seja falso:
        - **rGraph[u][v]** = (0,0);

# Cenário 1.2

## CAMINHOS DE CAPACIDADE MÁXIMA

ATRÁS EXPLICADO

## EDMOND-KARP

De seguida, enquanto a função **bfs(Breadth-First Search)** retornar verdadeiro, faz-se:

- **path\_flow** = **INT\_MAX**;
- para **v** = **D**, enquanto **v** != **O**, faz:
  - **u** = **P[v]**;
  - **path\_flow** = min(**path\_flow**, **rGraph[u][v].first**);
  - **v** = **P[v]**;
- **Path.second** = **INT\_MAX**;
- Empurra para trás de **Path.first** o nó **D**;
- para **v** = **D**, enquanto **v** != **O**, faz:
  - **u** = **P[v]**;
  - **rGraph[u][v].first** -= **path\_flow**;
  - **rGraph[v][u].first** -= **path\_flow**;
  - **Path.second** = min (**Path.second**, **rGraph[u][v].second**);
  - Empurra para trás de **Path.first** o nó **u**;
- Empurra para trás de **AllPaths** o par de valores constituídos em **Path**;

# Cenário 1.2

## CAMINHOS DE CAPACIDADE MÁXIMA

ATRÁS EXPLICADO

### EDMOND-KARP

Os caminhos possíveis que poderão ser utilizados no percurso de **O** até **D** estão guardados no Vetor dinâmico **AllPaths**, constituído por um par de valores: um vetor constituído com os nós utilizados num percurso e um valor que representa a capacidade do percurso.

**Complexidade temporal:**  $O(|V| * |E|^2)$ , onde **V** corresponde ao número de nós e **E** ao número de arestas. Quanto à capacidade espacial, esta é  $O(2 * |V|^2 + |V|) = O(2 * |V|^2)$ .

## Cenário 1.2

CAMINHOS DE  
CAPACIDADE  
MÁXIMA

+

MÁXIMIZAÇÃO  
DE FLUXO

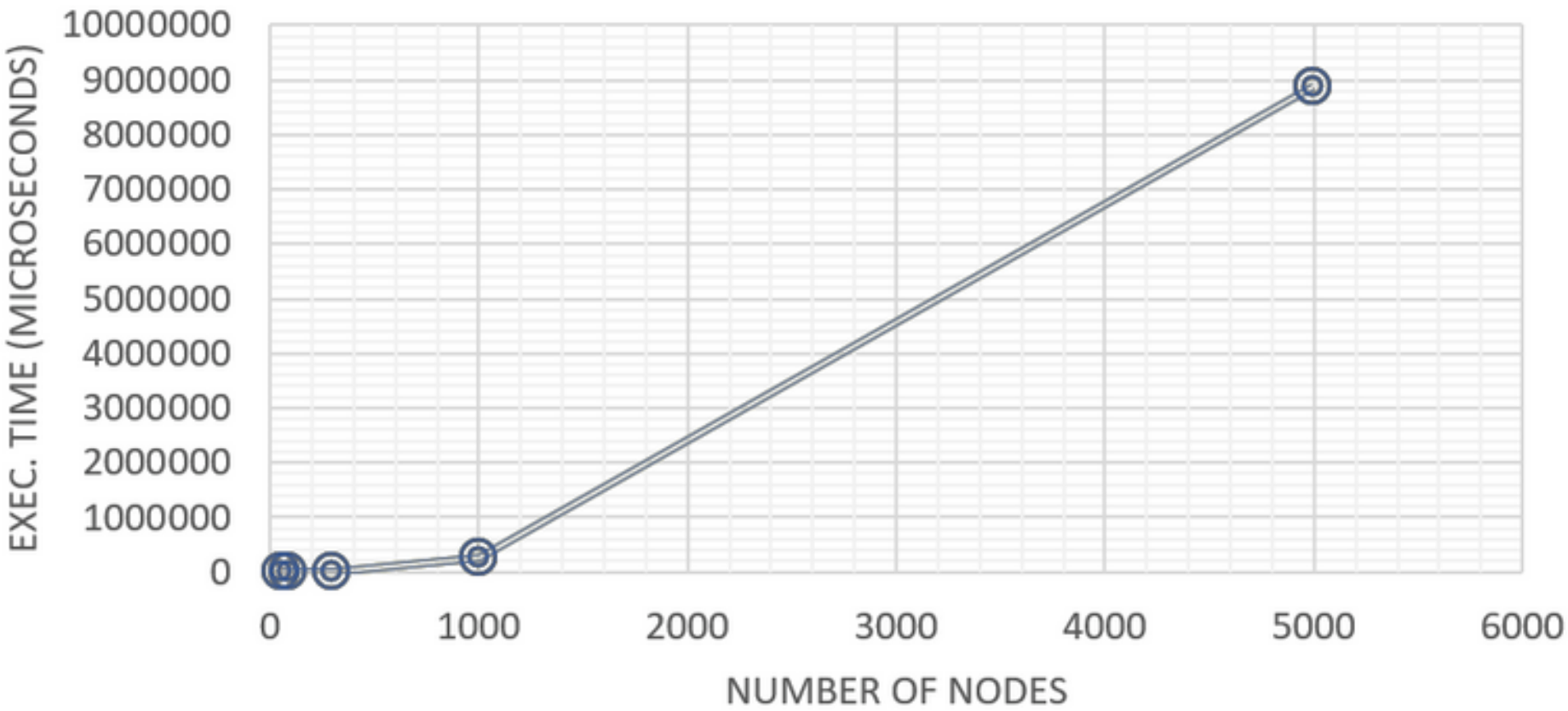
Finalmente, os caminhos obtidos em ambos algoritmos são comparados entre si de modo a apresentar as soluções não comparáveis entre os pontos **O** e **D**.

# Cenário 1.2

Avaliação empírica

Vertex	Edges	V * E	Tempo de exec. (µs)
50	136	6 800	421
90	257	23 130	1346
300	1417	425 100	20004
1000	7533	7 533 000	272948
5000	49487	247 435 000	8877706

Performance 1.2



## Resultados da Avaliação Empírica:

Como referido anteriormente trata-se de uma função quadrática( $O(|V| * |E|^2)$ ).

# Cenário 2.1

Determinar um encaminhamento para um grupo separável,  
dada a sua dimensão

## Variáveis relevantes

### Paragens

- origem (**o**,  $\mathbf{o} \in \mathbb{N}$ )
- destino (**d**,  $\mathbf{d} \in \mathbb{N}$ )

### Linhas

- capacidade  
(**cl**,  $\mathbf{cl} \in \mathbb{N}$ )

### Grupo

- dimensão (**s**,  $\mathbf{s} \in \mathbb{N}$ )

## Função objetivo

$$\begin{aligned} \mathbf{C} \geq \mathbf{s} \wedge \mathbf{C} = \text{Min}(\sum \mathbf{cl}(\mathbf{u}, \mathbf{v})) \wedge \\ (\mathbf{u} \in \mathbf{S} \wedge \mathbf{v} \in \mathbf{T}) \wedge \\ \mathbf{l} \in \{\mathbf{o}, \dots, \mathbf{d}\} \wedge \\ \mathbf{s} = \sum \mathbf{s}(\mathbf{o}, \mathbf{v}) = \sum \mathbf{s}(\mathbf{v}, \mathbf{d}) \wedge \\ \mathbf{v} \in \{\mathbf{o}, \dots, \mathbf{d}\} \end{aligned}$$

Para um conjunto de caminhos compostos por  $\{\mathbf{o}, \dots, \mathbf{d}\}$ , o objetivo seria encontrar um caminho de capacidade superior à dimensão do grupo ( $\mathbf{C} \geq \mathbf{s}$ ,  $\mathbf{C} \in \mathbb{N}$ ), sendo **C** igual à capacidade de um corte  $\{\mathbf{S}, \mathbf{T}\}$  mínimo. Obrigatoriamente, a dimensão do grupo que sai de **o** terá de ser a mesma que chega a **d**. Este problema poderá ter várias soluções de caminhos mas só uma será apresentada.

## Cenário 2.1

### EDMOND-KARP (MÁXIMIZAÇÃO DE FLUXO)

Utilizou-se a adaptação do algoritmo de **Edmond-Karp** para a **Maximização** de **Fluxo** (semelhante ao cenário 1.2), que permite descobrir os caminhos entre os pontos **O** e **D** de tal modo a que o fluxo seja máximo ou equivalente ao tamanho **s** dado à função.



# Cenário 2.1

## ALGORITMO EDMOND-KARPS

Tomando **AllPaths** como o vetor dinâmico constituído por múltiplos **Path**'s, que constitui um vetor dinâmico que contém um par de valores que representam, respetivamente, um vetor constituído pelo caminho entre **O** e **D**, **rGraph** como um vetor dinâmico de 2 dimensões que contém um par de valores que representam, respetivamente, o fluxo residual e a capacidade de cada aresta, **V** como o número de nós e **P** como um vetor de capacidade **V**, **E.dest** como o nó de destino de uma determinada aresta, **found** como um booleano, **INT\_MAX** como um valor infinitamente grande e **Max\_path\_flow** como um inteiro inicializado a 0.

Inicialmente, vai-se popular o vetor **rGraph**:

- Enquanto **u** < **V** (sendo que **u** inicializa-se a 0):
  - Enquanto **v** < **V**, (sendo que **v** inicializa-se a 0):
    - **found** = falso;
    - Para todos os ramos **E** do nó **u**, faz:
      - Caso **E.dest** seja igual a **v**, faz:
        - **found** = verdadeiro;
        - **rGraph[u][v]** = (E.capacity, E.capacity)
      - Caso **found** seja falso:
        - **rGraph[u][v]** = (0,0);

# Cenário 2.1

## ALGORITMO EDMOND-KARPS

De seguida, enquanto a função **bfs(Breadth-First Search)** retornar verdadeiro, faz-se:

- **path\_flow** = **INT\_MAX**;
- para **v** = **D**, enquanto **v** != **O**, faz:
  - **u** = **P[v]**;
  - **path\_flow** = min(**path\_flow**, **rGraph[u][v].first**);
  - **v** = **P[v]**;
- **Path.second** = **INT\_MAX**;
- Empurra para trás de **Path.first** o nó **D**;
- para **v** = **D**, enquanto **v** != **O**, faz:
  - **u** = **P[v]**;
  - **rGraph[u][v].first** -= **path\_flow**;
  - **rGraph[v][u].first** -= **path\_flow**;
  - **Path.second** = min (**Path.second**, **rGraph[u][v].second**);
  - Empurra para trás de **Path.first** o nó **u**;
- Empurra para trás de **AllPaths** o par de valores constituídos em **Path**;
- **Max\_path\_flow** += **path\_flow**;
- Caso **Max\_path\_flow** >= **s**, quebra o ciclo

## Cenário 2.1

### ALGORITMO EDMOND-KARPS

Os caminhos possíveis que poderão ser utilizados no percurso de **O** até **D** estão guardados no Vetor dinâmico **AllPaths**, constituído por um par de valores: um vetor constituído com os nós utilizados num percurso e um valor que representa a capacidade do percurso.

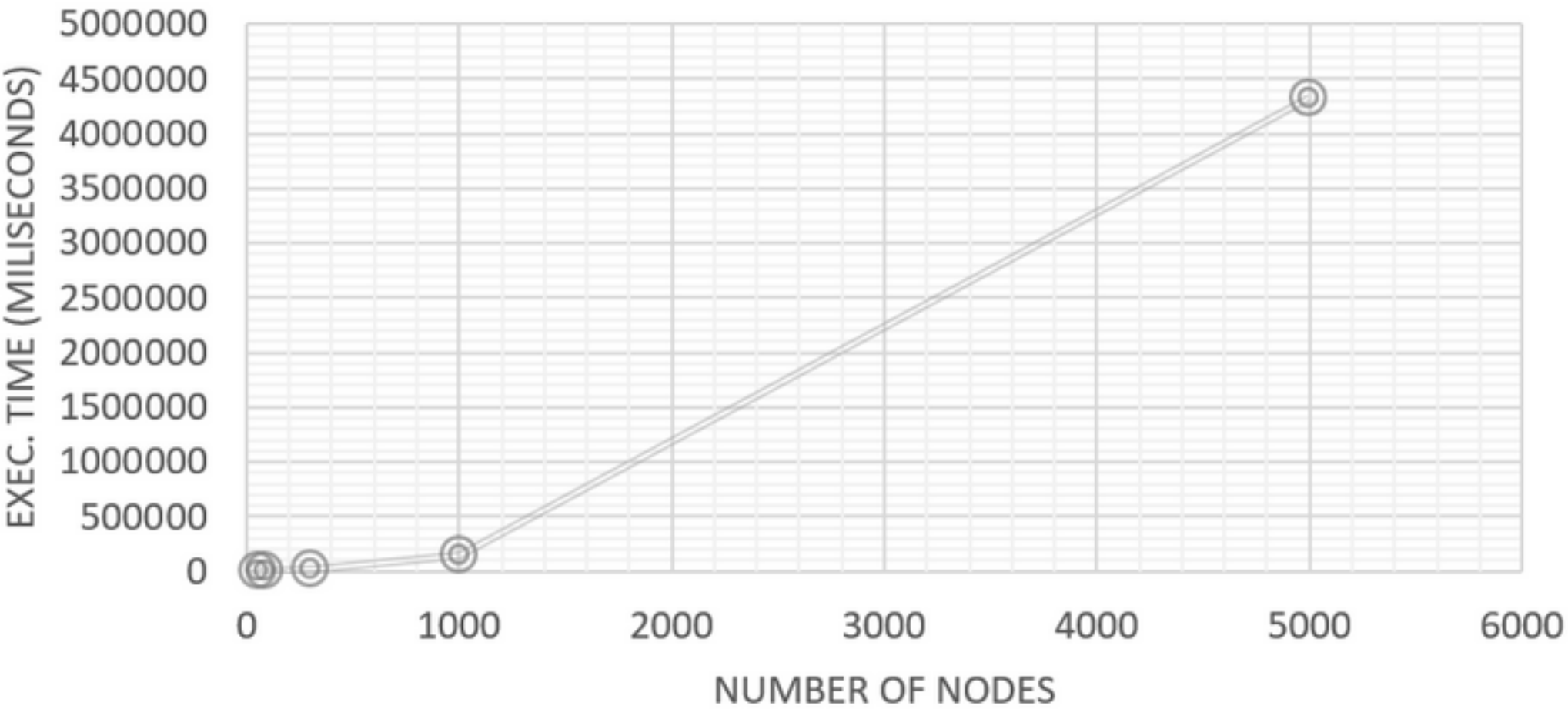
**Complexidade temporal:**  $O(|V| * |E|^2)$ , onde **V** corresponde ao número de nós e **E** ao número de arestas. Quanto à capacidade espacial, esta é  $O(2 * |V|^2 + |V|) = O(2 * |V|^2)$ .

# Cenário 2.1

Avaliação empírica

## Performance 2.1

Vertex	Edges	V * E	Tempo de exec. (μs)
50	136	6 800	389
90	257	23 130	1176
300	1417	425 100	14178
1000	7533	7 533 000	150250
5000	49487	247 435 000	4320984



### Resultados da Avaliação Empírica:

Como referido anteriormente trata-se de uma função quadrática( $O(|V| * |E|^2)$ ).

# Cenário 2.2

Corrigir um encaminhamento de um grupo separável, dado um aumento da sua dimensão

## Variáveis relevantes

### Paragens

- origem (**o**,  $\mathbf{o} \in \mathbb{N}$ )
- destino (**d**,  $\mathbf{d} \in \mathbb{N}$ )

### Linhas

- capacidade (**cl**,  $\mathbf{cl} \in \mathbb{N}$ )

### Grupo

- antiga dimensão (**os**,  $\mathbf{os} \in \mathbb{N}$ )
- nova dimensão (**ns**,  $\mathbf{ns} \in \mathbb{N}$ )

## Função objetivo

$$\begin{aligned} & ((\mathbf{ns} > \mathbf{C}) \rightarrow (\mathbf{C} \geq \mathbf{ns} \wedge \\ & \mathbf{C} = \text{Min}(\sum \mathbf{cl}(\mathbf{u}, \mathbf{v})) \wedge \\ & (\mathbf{u} \in \mathbf{S} \wedge \mathbf{v} \in \mathbf{T}) \wedge \\ & \mathbf{l} \in \{\mathbf{o}, \dots, \mathbf{d}\} \wedge \\ & \mathbf{ns} = \sum \mathbf{ns}(\mathbf{o}, \mathbf{v}) = \sum \mathbf{ns}(\mathbf{v}, \mathbf{d}) \wedge \\ & \mathbf{v} \in \{\mathbf{o}, \dots, \mathbf{d}\} )) \wedge \\ & (\neg(\mathbf{ns} > \mathbf{C}) \rightarrow (\mathbf{C} = \mathbf{C})) \end{aligned}$$

Para um conjunto de caminhos compostos por  $\{\mathbf{o}, \dots, \mathbf{d}\}$ , o objetivo seria encontrar um caminho de capacidade superior à nova dimensão do grupo ( $\mathbf{C} \geq \mathbf{ns}$ ,  $\mathbf{C} \in \mathbb{N}$ ), sendo **C** igual à capacidade de um corte  $\{\mathbf{S}, \mathbf{T}\}$  mínimo, caso seja necessário. Obrigatoriamente, a dimensão do grupo que sai de **o** terá de ser a mesma que chega a **d**. Este problema poderá ter várias soluções de caminhos mas só uma será apresentada.

## Cenário 2.2

### ALGORITMO EDMOND-KARPS

Para o cenário 2.1, fora guardado globalmente os vetores **P** e **rGraph**. Deste modo, é possível retomar o mesmo algoritmo, dentro do ciclo da função **bfs**, com as últimas instâncias guardadas.

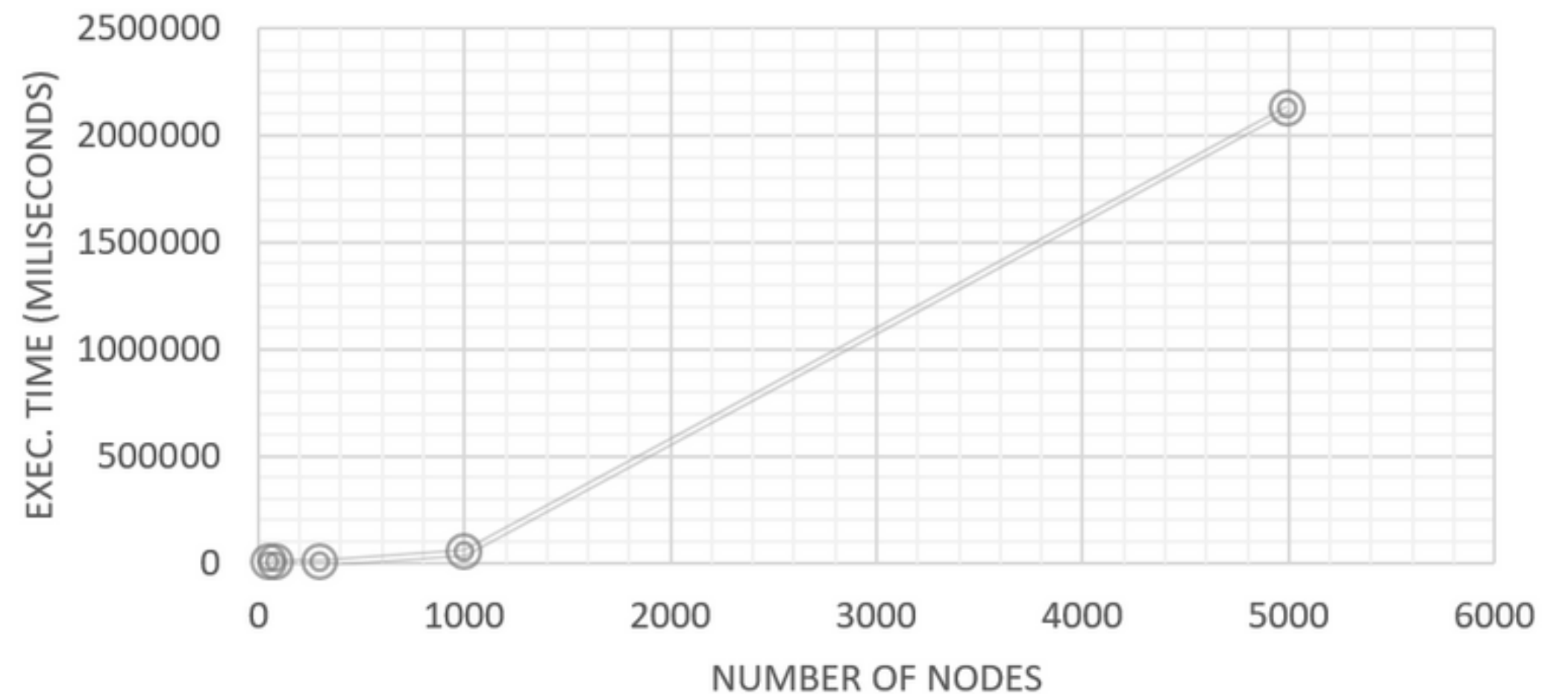
**Complexidade temporal:**  $O(|V| * |E|^2)$  no pior dos casos, onde **V** corresponde ao número de nós e **E** ao número de arestas. Quanto à capacidade espacial, esta é  $O(|V|^2 + |V|) = O(|V|^2)$ .

# Cenário 2.2

Avaliação empírica

## Performance 2.2

Vertex	Edges	V * E	Tempo de exec. (μs)
50	136	6 800	232
90	257	23 130	815
300	1417	425 100	2726
1000	7533	7 533 000	46510
5000	49487	247 435 000	2125886



### Resultados da Avaliação Empírica:

Como referido anteriormente trata-se de uma função quadrática( $O(|V| * |E|^2)$ ).



# Cenário 2.3

Maximizar a dimensão de um grupo separável

## Variáveis relevantes

### Paragens

- origem (**o**,  $\mathbf{o} \in \mathbb{N}$ )
- destino (**d**,  $\mathbf{d} \in \mathbb{N}$ )

### Linhas

- capacidade (**cl**,  $\mathbf{cl} \in \mathbb{N}$ )

## Função objetivo

$$\begin{aligned} \text{Max } \mathbf{C} = & \text{Min}(\sum \mathbf{cl}(\mathbf{u}, \mathbf{v}) \wedge \\ & (\mathbf{u} \in \mathbf{S} \wedge \mathbf{v} \in \mathbf{T}) \wedge \\ & \mathbf{l} \in \{\mathbf{o}, \dots, \mathbf{d}\} \wedge \\ \mathbf{C} = & \sum \mathbf{C}(\mathbf{o}, \mathbf{v}) = \sum \mathbf{C}(\mathbf{v}, \mathbf{d}) \wedge \\ & \mathbf{v} \in \{\mathbf{o}, \dots, \mathbf{d}\} \end{aligned}$$

Para um conjunto de caminhos compostos por  $\{\mathbf{o}, \dots, \mathbf{d}\}$ , o objetivo seria encontrar o caminho de maior capacidade (**C**,  $\mathbf{C} \in \mathbb{N}$ ), sendo **C** igual à capacidade de um corte  $\{\mathbf{S}, \mathbf{T}\}$  mínimo. Obrigatoriamente, a dimensão do grupo que sai de **o** terá de ser a mesma que chega a **d**. Este problema poderá ter várias soluções de caminhos mas só uma será apresentada.



## Cenário 2.3

### ALGORITMO EDMOND-KARPS

Semelhante ao cenário 2.1, dentro do ciclo da função **bfs** ignora-se a linha em que se verifica se a variável **Path\_max\_flow** é igual ou superior o **s**.

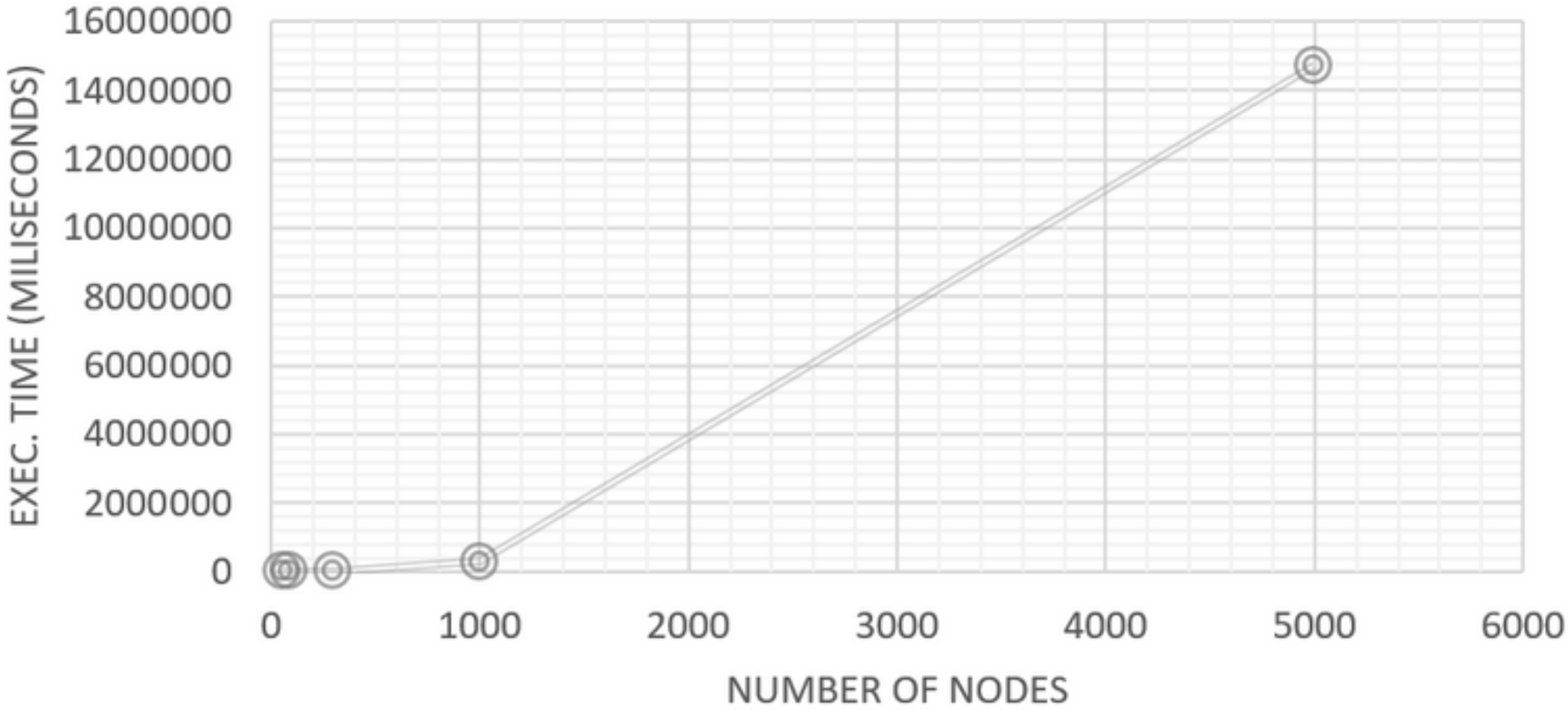
**Complexidade temporal:**  $O(|V| * |E|^2)$  no pior dos casos, onde **V** corresponde ao número de nós e **E** ao número de arestas. Quanto à capacidade espacial, esta é  $O(|V|^2 + |V|) = O(|V|^2)$ .

# Cenário 2.3

Avaliação empírica

## Performance 2.3

Vertex	Edges	V * E	Tempo de exec. (µs)
50	136	6 800	447
90	257	23 130	1235
300	1417	425 100	16254
1000	7533	7 533 000	290038
5000	49487	247 435 000	14736839



### Resultados da Avaliação Empírica:

Como referido anteriormente trata-se de uma função quadrática( $O(|V| * |E|^2)$ ).

# Cenário 2.4

Determinar a duração máxima de viagem de um grupo separável.

## Variáveis relevantes

### Paragens

- origem (**o**,  $\mathbf{o} \in \mathbb{N}$ )
- destino (**d**,  $\mathbf{d} \in \mathbb{N}$ )

### Linhas

- duração (**DI**,  $\mathbf{DI} \in \mathbb{N}$ )

## Função objetivo

$$\text{Max } \mathbf{T} = \Sigma \mathbf{DI}(\mathbf{o}, \mathbf{d})$$

Para um caminho  $\{\mathbf{o}, \dots, \mathbf{d}\}$ , o objetivo seria determinar o período de tempo (**T**,  $\mathbf{T} \in \mathbb{N}$ ) necessário para a totalidade do grupo chegar ao destino **d**, sendo **T** igual à duração do caminho mais demorado entre **o** e **d**.

## Cenário 2.4

### MÉTODO DO CAMINHO CRÍTICO (EARLIEST START)

É utilizado inicialmente o algoritmo de **Edmond-Karp** (utilizado nos cenários anteriores) para construir um grafo auxiliar que contém o mesmo número de nós e, somente, as arestas utilizadas no percurso entre **o** e **d**.

De seguida, é utilizado o **Método** do **Caminho Crítico** (**Earliest Start**) para determinar a duração mínima do percurso entre **o** e **d**.

**Complexidade temporal:**  $O(|V| * |E|^2 + |V| * |E| + 2 * |V|) = O(|V| * |E|^2)$ , onde **V** corresponde ao número de nós e **E** ao número de arestas. Quanto à capacidade espacial, esta é  $O(|V|^2 + |V|) = O(|V|^2)$ .

## Cenário 2.4

### MÉTODO DO CAMINHO CRÍTICO (EARLIEST START)

Tomando **S** como uma fila de valores inteiros, **G** como um grafo constituído por um vetor de nós, **v** como um nó de **G**, **v.ES** como a data mais próxima para os caminhos que iniciam em **v**, **v.parent** como o nó que o precede, **v.eDegree** como o grau do nó, **v.id** como o número do nó **v**, **minDuration** como um inteiro inicializado a -1, **E.dest** como o nó de destino da aresta **E** e **E.dur** como a duração da aresta **E**;

Para cada nó **v** do grafo **G**, fazer:

- **v.ES** = 0;
- **v.parent** = 0;
- **v.eDegree** = 0;

De seguida, para cada nó **v** do grafo **G**, fazer:

- Para toda a aresta **E** do nó, fazer:
  - **G[E.dest].eDegree** += 1;

Para cada nó **v** do grafo **G**, fazer:

- Caso **v.eDegree** = 0, adiciona-se **v.id** para a fila **S**;

## Cenário 2.4

### MÉTODO DO CAMINHO CRÍTICO (EARLIEST START)

Enquanto **S** não estiver vazia, fazer:

- $u$  = elemento da frente de **S**;
- remover o elemento da frente de **S**;
- **minDuration** =  $\max(\mathbf{minDuration}, \mathbf{G}[u].ES)$ ;
- Para todas as arestas **E** de **u**, fazer:
  - $\mathbf{G}[E.dest].ES = \max(\mathbf{G}[E.dest].ES, \mathbf{G}[u].ES + E.dur)$ ;
  - $\mathbf{G}[E.dest].parent = u$ ;
- $\mathbf{G}[E.dest].eDegree -= 1$ ;
- Introduzir na fila **E.dest** caso  $\mathbf{G}[E.dest].eDegree = 0$ ;

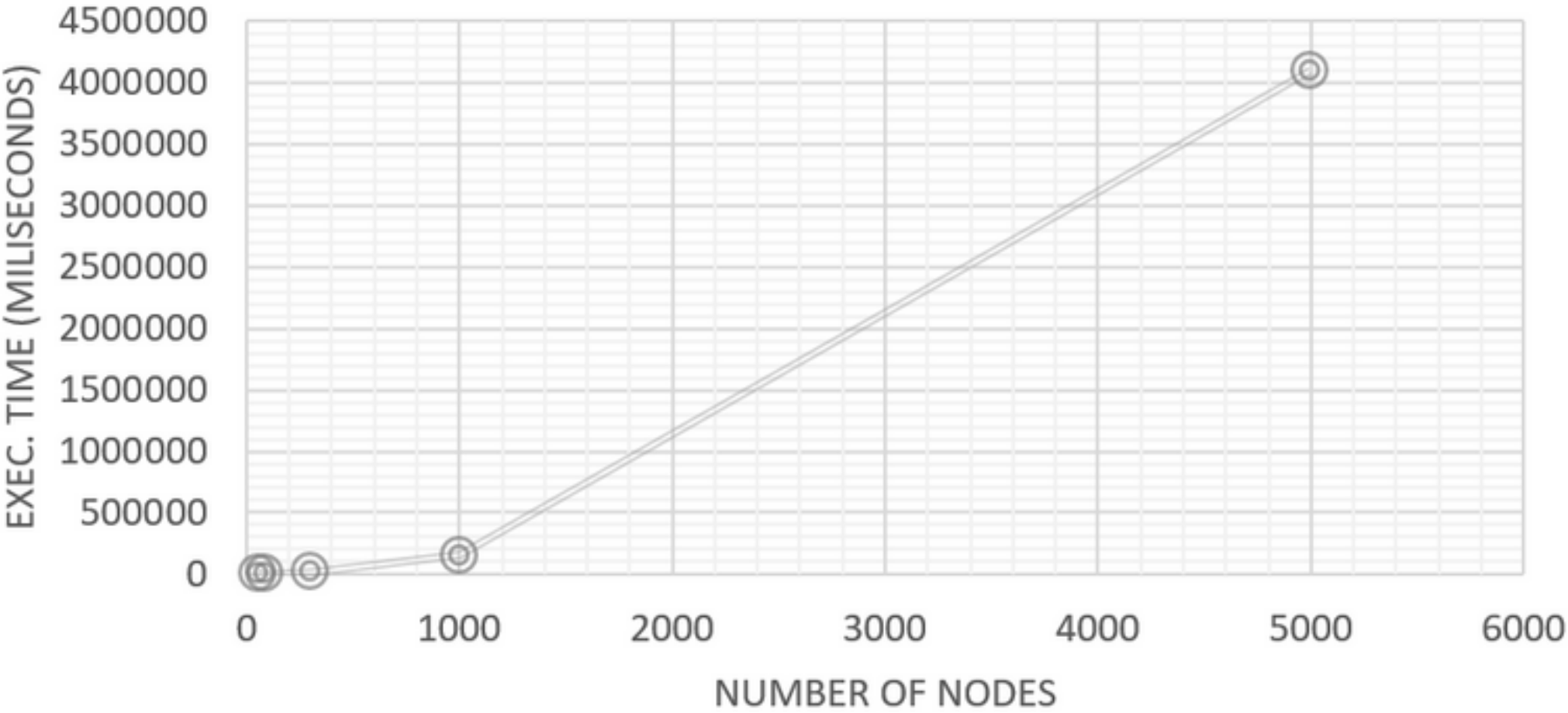
**Complexidade temporal:**  $O(|V| * |E|^2 + |E| + 2 * |V|) = O(|V| * |E|^2)$ , onde **V** corresponde ao número de nós e **E** ao número de arestas. Quanto à capacidade espacial, esta é  $O(|V|^2 + |V|) = O(|V|^2)$ .

# Cenário 2.4

Avaliação empírica

## Performance 2.4

Vertex	Edges	V * E	Tempo de exec. (µs)
50	136	6 800	414
90	257	23 130	1166
300	1417	425 100	12419
1000	7533	7 533 000	152090
5000	49487	247 435 000	4096887



### Resultados da Avaliação Empírica:

Como referido anteriormente trata-se de uma função quadrática( $O(|V| * |E|^2)$ ).

# Cenário 2.5

Determinar o tempo máximo de espera entre elementos de um grupo separável.

## Variáveis relevantes

### Paragens

- origem (**o**,  $\mathbf{o} \in \mathbb{N}$ )
- destino (**d**,  $\mathbf{d} \in \mathbb{N}$ )

### Linhas

- duração (**DI**,  $\mathbf{DI} \in \mathbb{N}$ )

## Função objetivo

$$\text{Max } \mathbf{T} = \text{Max}(\mathbf{DI}(\mathbf{o}, \mathbf{vi})) - \text{Max}(\mathbf{DI}(\mathbf{o}, \mathbf{vj})) - \Sigma(\mathbf{DI}(\mathbf{vi}, \mathbf{vj}))$$

Para um caminho  $\{\mathbf{o}, \dots, \mathbf{d}\}$ , o objetivo seria determinar o maior período de tempo (**T**,  $\mathbf{T} \in \mathbb{N}$ ) que parte do grupo aguardaria pelos restantes numa paragem (**v**,  $\mathbf{v} \in \mathbb{N}$ ) sendo **T** igual à maior das diferenças entre os caminhos de maior duração, menos a duração da viagem entre eles.



## Cenário 2.5

### MÉTODO DO CAMINHO CRÍTICO (LATEST FINISH)

É utilizado inicialmente o algoritmo de **Edmond-Karp** (utilizado nos cenários anteriores) para construir um grafo auxiliar que contém o mesmo número de nós e, somente, as arestas utilizadas no percurso entre **o** e **d**.

De seguida, é utilizado o **Método do Caminho Crítico "Earliest Start"** para preencher, em cada nó, o campo representativo da data de início mais próxima dos caminhos que se iniciam nele.

Por fim, é utilizado o **Método do Caminho Crítico "Latest Finish"** para comparar, entre caminhos, qual seria o intervalo máximo de espera e quais os nós em que tal aconteceria.

## Cenário 2.5

### MÉTODO DO CAMINHO CRÍTICO (LATEST FINISH)

Tomando **S** e **W** como dois vetores de valores inteiros de tamanho **V**, **G** como um grafo constituído por um vetor de nós, **v** como um nó de **G**, **v.ES** como a data mais próxima para os caminhos que iniciam em **v**, **v.id** como o número do nó **v**, **E.dest** como o nó de destino da aresta **E** e **E.dur** como a duração da aresta **E**, **maxW** inicializado a um número infinitamente pequeno;

Para cada nó **v** do grafo **G**, fazer:

- Para toda a aresta **E** do nó, fazer:
  - **maxDuration** = **G[E.dest].ES** - **G[v.id].ES** - **E.dur**;
  - **W[E.dest]** = max(**W[E.dest]**, **maxDuration**);

De seguida, para cada elemento **w** do vetor **W**, fazer:

- **maxW** = max(**maxWait**, **w**);

Enquanto **i** for menor que o tamanho do vetor **W**:

- caso **W[i]** == **maxW**, introduzir na fila **S i**;

## Cenário 2.5

### MÉTODO DO CAMINHO CRÍTICO (LATEST FINISH)

Caso **maxW** seja um valor positivo, os valores guardados no vetor **S** representam os nós nos quais será necessário esperar por um outro caminho para se realizar.

Caso o valor seja 0, então todos os caminhos que realizam o percurso **o** a **d** chegarão simultaneamente.

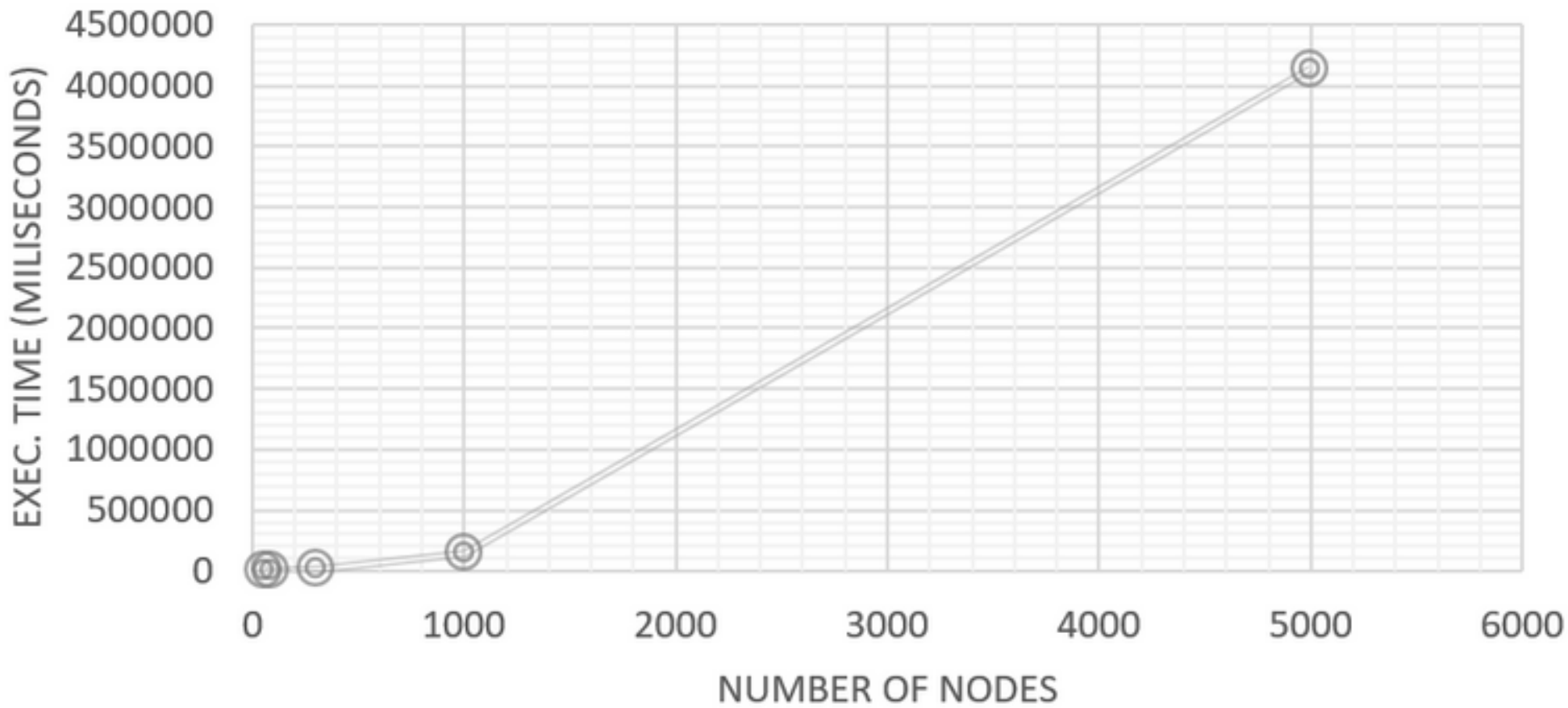
**Complexidade temporal:**  $O(|V| * |E|^2 + 2 * |V| * |E| + 4 * |V|) = O(|V| * |E|^2)$ , onde **V** corresponde ao número de nós e **E** ao número de arestas. Quanto à capacidade espacial, esta é  $O(|V|^2 + 3 * |V|) = O(|V|^2)$ .

# Cenário 2.5

Avaliação empírica

Performance 2.5

Vertex	Edges	V * E	Tempo de exec. (μs)
50	136	6 800	414
90	257	23 130	1176
300	1417	425 100	12247
1000	7533	7 533 000	144429
5000	49487	247 435 000	4144612



**Resultados da Avaliação Empírica:**

Como referido anteriormente trata-se de uma função quadrática( $O(|V| * |E|^2)$ ).

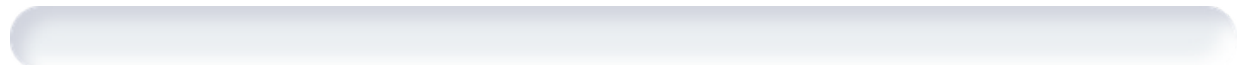
# Destaque de Algoritmo

Edmond-Karp

Destaca-se o algoritmo de Edmond-Karp que fora utilizado com maior frequência ao longo dos cenários.

Com ligeira adaptação do código, foi possível fornecer os resultados mais ótimos dentro de um intervalo de tempo aceitável, especialmente no cenário 1.2.

Salienta-se o facto de se recorrer a um vetor dinâmico para guardar o fluxo residual, o que permitiu otimizar o tempo decorrido evitando redundâncias desnecessárias.



## Principal dificuldade encontrada

## Esforço de cada elemento do grupo

### CENÁRIO 1.2

No cenário 1.2 foi particularmente difícil lidar com o grande número de caminhos possíveis. A implementação utilizada permitiu obter uma quantidade razoável de caminhos pareto-ótimos.

Henrique Silva 33%  
João Araújo 33%  
Vasco Guedes 33%