2º projeto DA

G73

up202008169 – André Lima

up202006137 – Guilherme Almeida

up202004260 – Mariana Lobão

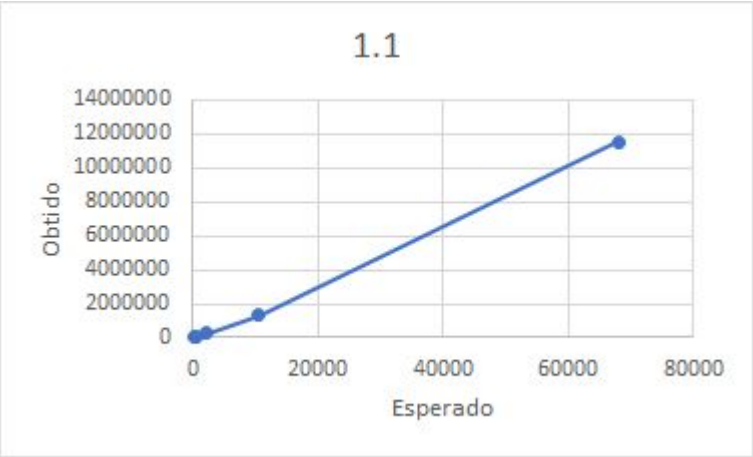
Descrição dos problemas

- O projeto consistia em, partindo de grafos variados, apresentar um algoritmo que permitisse a um grupo percorrê-lo em determinadas condições, sendo estas:
  - Pelo percurso de maior capacidade, não permitindo separações
  - Por percursos pareto-ótimos, não permitindo separações
  - Pelo percurso de maior capacidade, permitindo separações e calculando tempos de espera

1.1 – Maximizar dimensão do grupo

Pretende-se encontrar um caminho de capacidade máxima entre dois pontos de uma rede.

- Algoritmo utilizado – Algoritmo de Dijkstra
- Complexidade temporal – O(|E| + |V| log |V|)



1.1 – Formalização Matemática

Variáveis de entrada:

E - conjunto de arestas do grafo

start - o ponto de início

end - o ponto de fim

cap[i] - a capacidade da aresta i

Variáveis de saída:

C - caminho com maior capacidade

1.1 – Formalização Matemática

Encontrar um caminho, $C \subseteq E$, no grafo, tal que, para qualquer outro caminho $X \subseteq E$:

Restrições: $\exists_{A \in C}\ start \in A\ \wedge \exists_{A \in C}\ end \in A$

$\exists_{A \in X}\ start \in A\ \wedge \exists_{A \in X}\ end \in A$

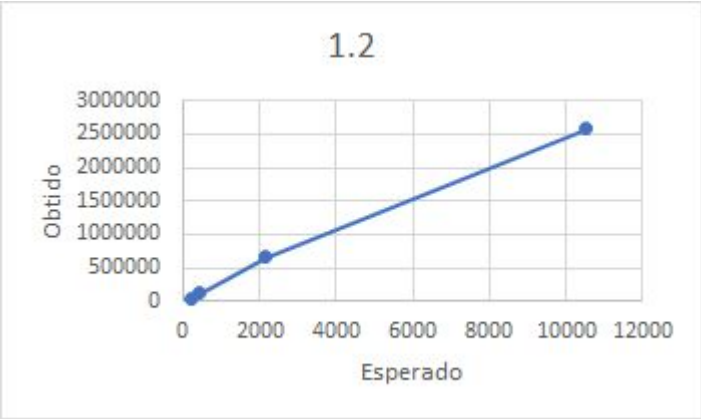
Satifazer: $\forall_{i \in C}\ min(cap[i]) \geq \forall_{j \in X}\ min(cap[j])$

1.2 – Caminhos pareto-ótimos

Pretende-se encontrar caminhos pareto-ótimos, em termos de capacidade e duração, entre dois pontos de uma rede.

Para isso, basta determinar o caminho com maior capacidade e menor duração e o caminho de menor duração e maior capacidade.

- Algoritmo utilizado – Algoritmo de Dijkstra
- Complexidade temporal – O(|E| + |V| log |V|)



1.2 – Formalização Matemática

Variáveis de entrada:

E - conjunto de arestas do grafo

start - o ponto de início

end - o ponto de fim

cap[i] - a capacidade da aresta i

dur[i] - a duração da aresta i

Variáveis de saída:

C_1 - caminho com maior capacidade e menor duração

C_2 - caminho com menor duração e maior capacidade

1.2 - Formalização Matemática

Encontrar 2 caminhos, $C_1,\ C_2 \subseteq E$, no grafo, tal que, para qualquer outro caminho $X \subseteq E$:

Restrições:

$$\exists_{A \in C_1}\ start \in A \land \exists_{A \in C_1}\ end \in A$$
$$\exists_{A \in C_2}\ start \in A \land \exists_{A \in C_2}\ end \in A$$
$$\exists_{A \in X}\ start \in A\ \land \exists_{A \in X}\ end \in A$$

Satisfazer:

$$\forall_{i \in C_1}\ min(cap[i]) \geq \forall_{j \in X}\ min(cap[j])$$
$$\forall_{i \in C_1}\ min(cap[i]) = \forall_{j \in X}\ min(cap[j]) \Rightarrow \#C_1 \leq \#X$$

$$\#C_2 \leq \#X$$
$$\#C_2 = \#X \Rightarrow \forall_{i \in C_2}\ min(cap[i]) \geq \forall_{j \in X}\ min(cap[i])$$

2.1 – Encontrar um encaminhamento, com separações

Dado o tamanho de um grupo, pretende-se encontrar um encaminhamento possível com capacidade pelo menos igual ao tamanho do grupo, tendo em conta que o grupo pode separar-se.

- Algoritmo utilizado – Algoritmo de Edmonds-Karp
- Complexidade temporal – $O(|E|^2 |V|)$

2.1 – Formalização Matemática

Variáveis de entrada:

E - conjunto de arestas do grafo

V - conjunto de nós do grafo

start - o ponto de início

end - o ponto de fim

cap[i] - a capacidade da aresta i

S - tamanho do grupo

Variáveis de saída:

F[i] - fluxo da aresta i

Fin[i] - fluxo que entra no nó i

Fout[i] - fluxo que sai do nó i

2.1 – Formalização Matemática

Encontrar um conjunto de fluxos , F , no grafo:

Restrições: $\forall_{1 \leq i \leq \#E} \; F[i] \leq cap[i]$

$$\forall_{n \in V} \; n \neq start \; \vee \; n \neq end \; \Rightarrow \; F_{in}[n] = F_{out}[n]$$

Satisfazer: $F_{out}[start] = F_{in}[end] \geq S$

2.2 – Permitir aumento da dimensão do grupo, corrigir encaminhamento

Dado um encaminhamento prévio, pretende-se aumentar a capacidade desse mesmo encaminhamento num dado número de unidades.

- Algoritmo utilizado – Algoritmo de Edmonds-Karp
- Complexidade temporal – $O(|E|^2 |V|)$

2.2 – Formalização Matemática

Variáveis de entrada:

E - conjunto de arestas do grafo

V - conjunto de nós do grafo

start - o ponto de início

end - o ponto de fim

cap[i] - a capacidade da aresta i

X[i] - fluxo da aresta i, no encaminhamento inicial

Xin[i] - fluxo que entra no nó i, no encaminhamento inicial

Xout[i] - fluxo que sai do nó i, no encaminhamento inicial

I - aumento desejado do fluxo

Variáveis de saída:

F[i] - fluxo da aresta i

Fin[i] - fluxo que entra no nó i

Fout[i] - fluxo que sai do nó i

2.2 – Formalização Matemática

Encontrar um conjunto de fluxos , F , no grafo, tal que, para um conjunto de fluxos X :

Restrições:

$$\forall_{1 \leq i \leq \#E} \; F[i] \leq cap[i]$$

$$\forall_{1 \leq i \leq \#E} \; X[i] \leq cap[i]$$

$$\forall_{n \in V} \; n \neq start \; \vee \; n \neq end \; \Rightarrow \; X_{in}[n] = X_{out}[n]$$

$$\forall_{n \in V} \; n \neq start \; \vee \; n \neq end \; \Rightarrow \; F_{in}[n] = F_{out}[n]$$

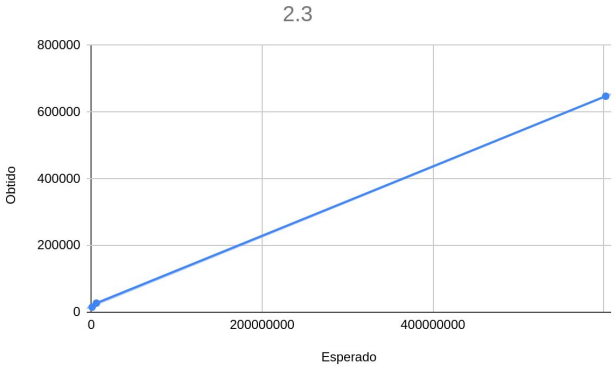
$$F_{out}[start] = F_{in}[end]$$

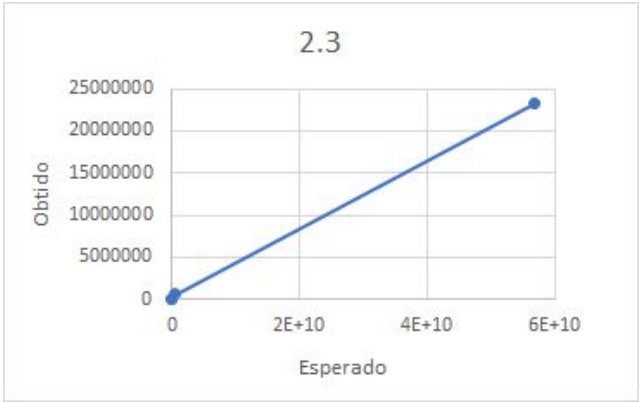
Satisfazer: $F_{out}[start] = X_{out}[start] + I$

2.3 – Maximizar a dimensão do grupo, com separações

Maximizar a capacidade do caminho que leva de um ponto a outro numa rede.

- Algoritmo utilizado – Algoritmo de Edmonds-Karp
- Complexidade temporal – $O(|E|^2 |V|)$





2.3 – Formalização Matemática

Variáveis de entrada:

E - conjunto de arestas do grafo

V - conjunto de nós do grafo

start - o ponto de início

end - o ponto de fim

cap[i] - a capacidade da aresta i

Variáveis de saída:

F[i] - fluxo da aresta i

Fin[i] - fluxo que entra no nó i

Fout[i] - fluxo que sai do nó i

2.3 – Formalização Matemática

Encontrar um conjunto de fluxos , F , no grafo:

Restrições:

$$\forall_{1 \leq i \leq \#E} \; F[i] \leq cap[i]$$

$$\forall_{n \in V} \; n \neq start \; \vee \; n \neq end \; \Rightarrow \; F_{in}[n] = F_{out}[n]$$

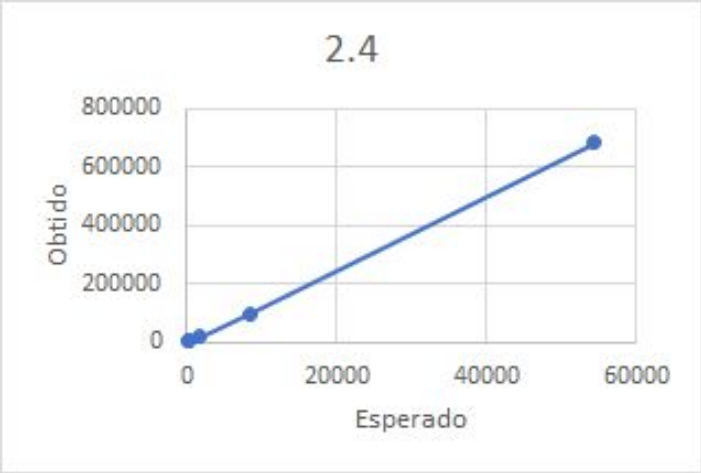
$$F_{out}[start] = F_{in}[end]$$

Maximizar: $F_{out}[start]$

2.4 – Encontrar o momento em que as pessoas se irão reencontrar

Dado um encaminhamento, pretende-se encontrar o momento em que as pessoas se irão reencontrar no destino.

- Algoritmo utilizado – Caminho Crítico (Earliest Start)
- Complexidade temporal – O(|E| + |V|)



2.4 – Formalização Matemática

Variáveis de entrada:

E - conjunto de arestas do grafo

V - conjunto de nós do grafo

start - o ponto de início

end - o ponto de fim

cap[i] - a capacidade da aresta i

dur[i] - a duração da aresta i

F[i] - fluxo da aresta i

Fin[i] - fluxo que entra no nó i

Fout[i] - fluxo que sai do nó i

Dest[i] - o ponto de destino da aresta i

Variáveis de saída:

EarliestStart[i] - tempo a partir do qual todos os grupos chegados ao nó i podem continuar a sua viagem

2.4 – Formalização Matemática

Encontrar um conjunto de fluxos , F :

Restrições: $\forall_{1 \leq i \leq \#E} F[i] \leq cap[i]$

$$\forall_{n \in V}\ n \neq start \ \vee \ n \neq end \ \Rightarrow \ F_{in}[n] = F_{out}[n]$$

$$F_{out}[start] = F_{in}[end]$$

$$\forall_{n \in V} \forall_{j \in Out[n]} F[j] > 0 \ \Rightarrow \ EarliestStart[n] + dur[j] \leq EarliestStart[Dest[j]]$$

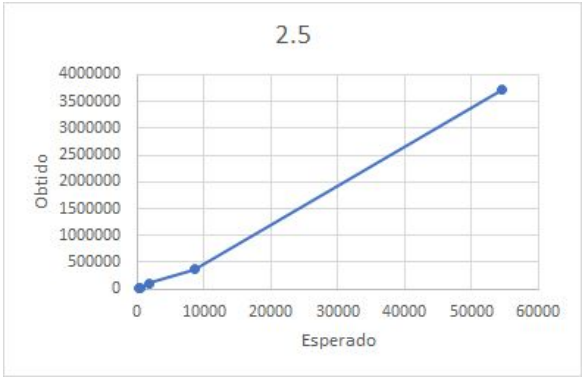
Minimizar: $EarliestStart[end]$

2.5 – Determinar os tempos de espera

Dado um encaminhamento, pretende-se determinar os tempos de espera em cada um dos pontos da rede, se existirem.

Para isso, basta determinar o Earliest Start de cada viagem na rede e, posteriormente, calcular, para cada nó, a diferença entre o Earliest Start desse nó e o instante em que o primeiro grupo chega a esse nó.

- Algoritmo utilizado – Caminho Crítico (Earliest Start)
- Complexidade temporal – O(|E| + |V|)



2.5 – Formalização Matemática

Variáveis de entrada:

E - conjunto de arestas do grafo

V - conjunto de nós do grafo

start - o ponto de início

end - o ponto de fim

cap[i] - a capacidade da aresta i

dur[i] - a duração da aresta i

F[i] - fluxo da aresta i

Fin[i] - fluxo que entra no nó i

Fout[i] - fluxo que sai do nó i

Origin[i] - o ponto de origem da aresta i

Dest[i] - o ponto de destino da aresta i

Variáveis de saída:

EarliestStart[i] - tempo a partir do qual todos os grupos chegados ao nó i podem continuar a sua viagem

Wait[i] - tempo de espera máximo no nó i

2.5 – Formalização Matemática

Encontrar um conjunto de fluxos , F :

Restrições:

$$\forall_{1 \leq i \leq \#E} \; F[i] \leq cap[i]$$

$$\forall_{n \in V} \; n \neq start \; \vee \; n \neq end \; \Rightarrow \; F_{in}[n] = F_{out}[n]$$

$$F_{out}[start] = F_{in}[end]$$

$$\forall_{n \in V} \; \forall_{j \in Out[n]} \; F[j] > 0 \; \Rightarrow \; EarliestStart[n] + dur[j] \leq EarliestStart[Dest[j]]$$

$$\forall_{n \in V} \; \forall_{j \in In[n]} \; F[j] > 0 \Rightarrow Wait[n] \geq EarliestStart[n] - (EarliestStart[Origin[j]] + dur[j])$$

Minimizar: $\forall_{n \in V} \; Wait[n]$

Dificuldades e Participação

De um modo geral, consideramos que não houve nenhuma dificuldade que valha a pena mencionar.

Consideramos que todos os membros do grupo tiveram a mesma participação no desenvolvimento do projeto.

Exemplos e live demo

```
Choose one of the following scenarios:

[1] [Scenario 1] Groups that don't separate
[2] [Scenario 2] Groups that separate
[3] [Options] Change the dataset file

[4] Exit

? Your option [1 - 4]:
```

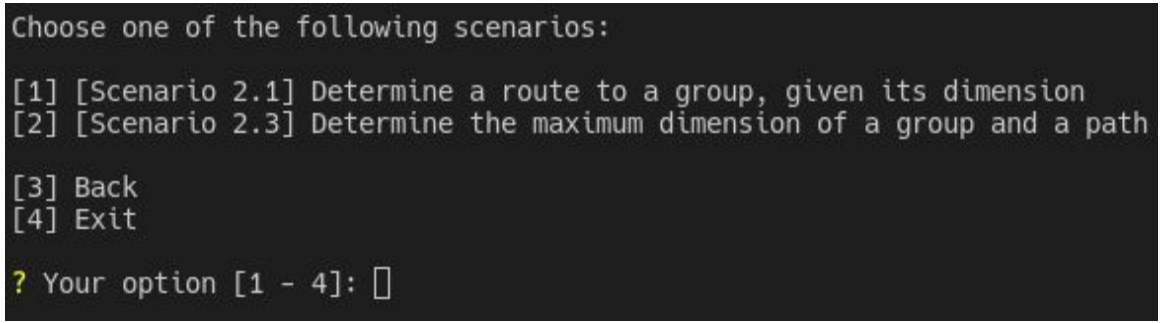
```
Choose one of the following scenarios:

[1] [Scenario 1] Maximize the size of the group and indicate any routing
[2] [Scenario 2] Find pareto-optimal paths

[3] Exit

? Your option [1 - 3]:
```

Exemplos e live demo



Exemplos e live demo

```
Choose one of the following scenarios:

[1] [Scenario 2.1] Determine a route to a group, given its dimension
[2] [Scenario 2.3] Determine the maximum dimension of a group and a path

[3] Back
[4] Exit

✓ Your option [1 - 4]: 1

✓ Choose a starting node: 8
✓ Choose an ending node: 12
✓ Choose the group size: 13

Format: [ origin -> destination / flow ]

[ 2 -> 41 / 10 ]
[ 3 -> 18 / 1 ]
[ 3 -> 23 / 9 ]
[ 8 -> 11 / 10 ]
[ 8 -> 46 / 4 ]
[ 11 -> 21 / 10 ]
[ 18 -> 2 / 1 ]
[ 21 -> 3 / 10 ]
[ 23 -> 2 / 9 ]
[ 41 -> 12 / 10 ]
[ 46 -> 12 / 4 ]

Dimension of the group: 14

Press ENTER to continue...
```

Exemplos e live demo

```
Choose one of the following scenarios:

[1] [Scenario 2.1] Determine a route to a group, given its dimension
[2] [Scenario 2.2] Correct a route, if needed, so that the size of the group can increase from a number of units given
[3] [Scenario 2.3] Determine the maximum dimension of a group and a path
[4] [Scenario 2.4] Determine when the group would meet again at the destination
[5] [Scenario 2.5] Determine the maximum waiting time and the places where there would be elements that wait for that time, assuming that the elements th
at leave the same location depart from that location at the same time (and as soon as possible),

[6] Back
[7] Exit

✓ Your option [1 - 7]: 3

✓ Choose a starting node: 1
✓ Choose an ending node: 50

Format: [ origin -> destination / flow ]

[ 1 -> 6 / 5 ]
[ 1 -> 38 / 5 ]
[ 3 -> 23 / 8 ]
[ 6 -> 9 / 2 ]
[ 6 -> 26 / 3 ]
[ 9 -> 33 / 2 ]
[ 11 -> 21 / 8 ]
[ 12 -> 50 / 1 ]
[ 13 -> 12 / 1 ]
[ 21 -> 3 / 8 ]
[ 23 -> 31 / 8 ]
[ 26 -> 11 / 8 ]
[ 31 -> 39 / 6 ]
[ 31 -> 50 / 2 ]
[ 33 -> 41 / 1 ]
[ 33 -> 45 / 1 ]
[ 38 -> 26 / 5 ]
[ 39 -> 50 / 6 ]
[ 41 -> 50 / 1 ]
[ 45 -> 13 / 1 ]

Max dimension of the group: 10

Press ENTER to continue...
```