

# Meat Wagons - Transporte de Prisioneiros

## Turma 2 Grupo 3

up201806250@fe.up.pt

Diogo Samuel Gonçalves Fernandes

up201806490@fe.up.pt

Hugo Miguel Monteiro Guimarães

up201806554@fe.up.pt

Telmo Alexandre Espirito Santo Baptista

18 de Abril de 2020

Projeto CAL - 2019/20 - MIEIC

Professor das Aulas Práticas: Rosaldo José Fernandes Rossetti



# Índice

<b>1</b>	<b>Descrição do Problema</b>	<b>3</b>
<b>2</b>	<b>Formalização do Problema</b>	<b>4</b>
2.1	Dados de Entrada . . . . .	4
2.2	Dados de Saída . . . . .	5
2.3	Restrições . . . . .	5
2.4	Função objetivo . . . . .	6
<b>3</b>	<b>Perspectiva de solução</b>	<b>7</b>
3.1	Pré-processamento dos dados de entrada . . . . .	7
3.2	Identificação do problema . . . . .	7
3.3	Caminho mais curto . . . . .	8
3.4	TSP/VRP . . . . .	8
<b>4</b>	<b>Bibliografia</b>	<b>9</b>

# 1 Descrição do Problema

Os transportes de prisioneiros entre diversos estabelecimentos como, por exemplo, as prisões, esquadras e tribunais são feitos usando carrinhas que se encontram adaptadas ao serviço. Estes veículos têm a necessidade de serem altamente resistentes uma vez que é necessário garantir que os prisioneiros não conseguem escapar.

Para este projeto, queremos otimizar o percurso dos veículos de forma a recolher e entregar os prisioneiros nos pontos de interesse. De modo a cumprir o pretendido, é possível dividir nas seguintes fases:

## **Primeira Iteração - Recolha não seletiva de prisioneiros utilizando uma única carrinha**

Inicialmente considere que só existe uma única camioneta para realizar todos os serviços. Com a primeira iteração pretende-se que apenas uma carrinha vá recolher os prisioneiros a uma dada localização, tendo em conta a urgência da situação. As situações que sejam mais exigentes serão respondidas primeiro pela carrinha.

É importante de notar que a recolha só pode ser efetuada se existirem caminhos que liguem todos os pontos de interesse, ou seja, o grafo necessita de ser conexo.

Algumas vezes, obras nas vias públicas podem fazer com que certas zonas tornem-se inacessíveis, inviabilizando o acesso ao destino de alguns prisioneiros. Avalie a conectividade do grafo, a fim de identificar pontos de recolha e de entrega com pouca acessibilidade.

## **Segunda Iteração - Recolha seletiva de prisioneiros utilizando uma única carrinha**

Durante a segunda fase, cada prisioneiro irá ser agrupado com outros prisioneiros sempre que seja possível, de modo a não exceder a capacidade da carrinha.

## **Terceira Iteração - Recolha seletiva de prisioneiros utilizando várias carrinhas**

Concluindo, nesta última fase vai-se ter em consideração o diverso número de carrinhas que a frota possui. Algumas carrinhas vão diferir de outras, tendo cada carrinha uma determinada função. Por exemplo, vão existir carrinhas específicas para transportar prisioneiros até aos aeroportos, linhas de comboio.

## 2 Formalização do Problema

### 2.1 Dados de Entrada

$C_i$  - sequência de veículos, sendo  $C_i(i)$  o seu  $i$ -ésimo elemento. Cada veículo é caracterizado por:

- *capacity* - número de prisioneiros que pode transportar
- *type* - tipo de veículo

$R_i$  - sequência de pedidos de transporte de prisioneiros, sendo  $R_i(i)$  o seu  $i$ -ésimo elemento. Cada pedido é caracterizado por:

- *pickup* - local de recolha dos prisioneiros
- *dest* - local de destino dos prisioneiros
- *numPris* - número de prisioneiros a serem transportados
- *type* - tipo de prisioneiros
- $p_d$  - peso da distância no trajeto a efetuar
- $p_t$  - peso do tempo no trajeto a efetuar

$G_i = (V_i, E_i)$  - grafo dirigido pesado, composto por:

- $V$  - vértices, representando pontos da rede viária, com:
  - $ID$  - Identificador único do vértice
  - $D$  - Densidade populacional no vértice
  - $Adj \subseteq E$  - arestas que saem do vértice
- $E$  - arestas, representando conexão entre dois pontos da rede viária, com:
  - $ID$  - Identificador único da aresta
  - $W_d$  - peso da aresta em relação à distância (representa a distância entre os dois vértices)
  - $W_t$  - peso da aresta em relação ao tempo (representa o tempo médio que demora a percorrer a distância entre os dois vértices, considerando o tráfego normal naquela conexão da rede viária)
  - *open* - se a conexão entre os vértices está aberta, isto é, se a rua estiver cortada por alguma razão então não é possível utilizar esta conexão

$S$  - vértice da central

## 2.2 Dados de Saída

$G_f = (V_f, E_f)$  - grafo dirigido pesado, tendo  $V_f$  e  $E_f$  os mesmos atributos que  $V_i$  e  $E_i$ , excluindo atributos específicos do algoritmo utilizado

$C_f$  - sequência de veículos com os serviços a realizar, sendo  $C_f(i)$  o seu  $i$ -ésimo elemento. Cada veículo é caracterizado por:

- $S$  - sequência de serviços a realizar, sendo  $S(i)$  o seu  $i$ -ésimo elemento. Cada serviço é caracterizado por:
  - *emptySeats* - número de lugares vazios
  - $R_f$  - sequência de pedidos atendidos, sendo  $R_f(i)$  o seu  $i$ -ésimo elemento. Cada pedido atendido é caracterizado por:
    - \* *pickupHour* - hora de chegada ao local de recolha
    - \* *destHour* - hora de chegada ao local de destino
    - \*  $p_d$  - peso da distância no trajeto a efetuar
    - \*  $p_t$  - peso do tempo no trajeto a efetuar
  - $P = e \in E_i$  - sequência de arestas a percorrer, sendo  $P(i)$  o seu  $i$ -ésimo elemento
  - *dist* - distância percorrida no serviço
  - *startHour* - hora esperada de início do serviço
  - *endHour* - hora esperada de termino do serviço

## 2.3 Restrições

### Sobre os dados de entrada

- $\forall i \in [0, |C_i|[: \text{capacity}(C_i(i)) > 0$ , uma vez que não faz sentido os veículos não poderem transportar prisioneiros
- $\forall r \in R_i, \text{dest}(r)$  deve pertencer ao mesmo componente fortemente conexo do grafo  $G_i$  que o vértice  $S$ , uma vez que o veículo tem de ser capaz de voltar à central
- $\forall r \in R_i, \text{numPris}(r) > 0$ , uma vez que não faz sentido ter um pedido para transportar zero prisioneiros
- $\forall r \in R_i, p_d \geq 0 \wedge p_t \geq 0 \wedge (p_d \neq 0 \vee p_t \neq 0)$
- $\forall e \in E_i, W_d(e) > 0 \wedge W_t(e) > 0$ , uma vez que o peso da aresta representa a distância ou o tempo médio necessário para percorrer a aresta, se esta distância ou tempo forem zero estaremos num ciclo no mesmo vértice
- $\forall e \in E_i, e$  deve ser uma rua ao qual os veículos possam utilizar, ruas que os veículos não tenham permissão para entrar não são incluídas no grafo  $G_i$
- $S \in V_i$ , uma vez que a central é um vértice do grafo  $G_i$

## Sobre os dados de saída

- $|C_f| \leq |C_i|$  - não se pode usar mais veículos que os disponíveis
- $\forall v_f \in V_f, \exists v_i \in V_i$  tal que  $v_i$  e  $v_f$  têm os mesmos valores para todos os atributos, com exceção de atributos específicos aos algoritmos utilizados
- $\forall e_f \in E_f, \exists e_i \in E_i$  tal que  $e_i$  e  $e_f$  têm os mesmo valores para todos os atributos, com exceção de atributos específicos aos algoritmos utilizados
- $\forall r_f \in R_f, \exists r_i \in R_i$  tal que  $r_f$  e  $r_i$  têm os mesmo valores para os atributos  $p_d$  e  $p_t$
- $\forall c \in C_f, \forall s \in S(c), 0 \leq \text{emptySeats} < \text{capacity}(c)$  pois cada serviço deve ter pelo menos um prisioneiro, e não pode haver sobrelotação do veículo
- $\forall c \in C_f, \forall s \in S(c), |R_f(s)| > 0$  uma vez que só faz sentido realizar um serviço se for houver um pedido de transporte de prisioneiros
- $\forall c \in C_f, \forall s \in S(c), \text{endHour}(s) > \text{startHour}(s)$
- $\forall c \in C_f, \forall s \in S(c), \text{startHour}(s) < \text{pickupHour}(\forall r \in R_f) < \text{endHour}(s) \wedge \text{startHour}(s) < \text{destHour}(\forall r \in R_f) \leq \text{endHour}(s)$

## 2.4 Função objetivo

A solução ótima passa por minizar a soma ponderada da distância percorrida e o tempo do serviço de uma determinada carrinha, que resulta na seguinte função:

$$\sum_{c \in C_f} \sum_{s \in S} \sum_{e \in P} (W_d(e) * \max(p_d(R_f(s))) + W_t(e) * \max(p_t(R_f(s)))$$

- $\max(p_d(R_f(s)))$  - é o maior valor para o peso da distância numa determinada sequência de pedidos de um serviço de uma carrinha
- $\max(p_t(R_f(s)))$  - é o maior valor para o peso do tempo numa determinada sequência de pedidos de um serviço de uma carrinha

Deste modo, obtivemos a função objetivo para o nosso problema que se encontra acima.

## 3 Perspectiva de solução

### 3.1 Pré-processamento dos dados de entrada

#### Grafo

Partindo da central todos as arestas que não forem alcançáveis têm a variável *open* definida como falsa. Além disso, todas os vértices do grafo que não pertencerem à componente fortemente conexa de origem devem ter as arestas que lhe alcançam marcadas como inacessíveis.

#### Pedidos de transporte de prisioneiros

Remover todos os pedidos de transporte de prisioneiros que não pertençam ao grafo pré-processado, isto é, remover aqueles que façam parte de arestas que têm a componente *open* definida como falsa.

Também devemos organizar os pedidos de transporte de prisioneiros por ordem decrescente do número de prisioneiros a transportar, facilitando depois no alocamento de veículos para o seu transporte.

#### Veículos para transporte de prisioneiros

Relativamente ao pré-processamento dos veículos de transporte, devemos organizá-los por ordem decrescente de capacidade. Assim, como também temos os pedidos de transporte de prisioneiros organizados por ordem decrescente do número de prisioneiros a transportar podemos potencialmente minizar o número de veículos utilizados.

### 3.2 Identificação do problema

A empresa de transporte de prisioneiros Meat Wagons necessita de transportar os prisioneiros de um ponto de recolha até um determinado destino. De modo a otimizar este transporte, a empresa optou por procurar o caminho mais eficiente para a efetuar a viagem.

Na primeira iteração, onde apenas está disponível um veículo, que realiza os pedidos de transporte um de cada vez, este problema trata-se do **caminho mais curto** entre a origem e o local de recolha seguido do **caminho mais curto** entre o local de recolha e o destino. A segunda iteração é semelhante à primeira iteração, variando apenas o número de veículos disponíveis para realizar os pedidos.

Na terceira e última iteração, não só varia o número de veículos disponíveis, como também o número de pedidos de transporte que um veículo pode realizar num único serviço, equiparando-se ao problema designado por **Vehicle Routing Problem**, uma generalização do problema do **Travelling Salesman Problem**, um problema NP-difícil.

Vale também realçar que os veículos devem retornar para a central no fim

### 3.3 Caminho mais curto

Este é o problema referido na primeira e segunda iteração, e trata-se de encontrar o percurso mais curto e eficiente entre dois pontos, ou entre todos os pares de pontos do grafo.

#### Entre dois pontos

Entre os vários algoritmos que existem para calcular o caminho mais curto entre dois pontos destacam-se os seguintes algoritmos:

##### Algoritmo de Dijkstra

Este algoritmo foi concebido por Edsger W. Dijkstra e resolve problemas do caminho mais curto de uma única origem em grafos que possuam pesos não negativos.

Para poder aplicar este algoritmo é necessário que cada vértice guarde a seguinte informação:

- $W$  - custo mínimo até ao local da origem (combinação linear da distância e tempo, como visto na função objetivo)
- $path$  - vértice antecessor no caminho mais curto

O algoritmo de Dijkstra pode utilizar uma *priorityqueue* ou um *array* para inserir os novos vértices. No caso da *priorityqueue* possui complexidade de  $O(|V| + |E| \log |V|)$  enquanto que no caso do array é  $O(V^2)$

##### Algoritmo A\*

##### Algoritmo de Bellman-Ford

#### Entre todos os pares de pontos

##### Algoritmo de Floyd-Warshall

##### Johnson's??

### 3.4 TSP/VRP

#### Brute-force

#### Held-Karp

#### nearest neighbour

#### genetic



## 4 Bibliografia

- Apresentações fornecidas pelo professor Rosaldo José Fernandes Rossetti nas aulas teóricas da cadeira Conceção e Análise de Algoritmos
- Shortest Path Problem
- Dijkstra's Algorithm
- Traveling Salesman Problem
- Vehicle Routing Problem