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

17 de Abril de 2020

Projeto CAL - 2019/20 - MIEIC

Professor das Aulas Práticas: Rosaldo José Fernandes Rossetti

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Índice

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 optimizar o percurso dos veículos de forma a recolher e entregar os prisioneiros nos pontos de interesse. De modo a cumprir o pretendido, é possivel 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 exisrem 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 possivel, de modo a não exceder a capacidade da carrinha.

Terceira Iteração - Recolha seletiva de prisioneiros utilizando várias carrinhas

Concluindo, nesta ultima 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 especificas 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 caraterizado 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 caraterizado 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 saiem 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 caraterizado por:

- S - sequência de serviços a realizar, sendo $S(i)$ o seu i-ésimo elemento. Cada serviço é caraterizado 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 caraterizado 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
  - $P = e \; \epsilon \; E_i$ - sequência de arestas a percorrer, sendo $P(i)$ o seu i-ésimo elemento
  - $dist$ - distância percorrida no serviço
  - $startHour$ - hora esperada de ínicio do serviço
  - $endHour$ - hora esperada de termino do serviço

2.3 Restrições

Sobre os dados de entrada

- $\forall i \; \epsilon \; [0, |C_i|[: capacity(C_i(i)) > 0$, uma vez que não faz sentido os veículos não poderem transportar prisioneiros
- $\forall r \; \epsilon \; R_i, dest(r)$ 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
- $\forall r \; \epsilon \; R_i, numPris(r) > 0$, uma vez que não faz sentido ter um pedido para transportar zero prisioneiros
- $\forall e \; \epsilon \; E_i, W_d(e) > 0 \wedge W_t(e) > 0$, 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
- $\forall e \; \epsilon \; E_i, e$ 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
- $S \; \epsilon \; V_i$, 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 especificos 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 especificos 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
- $\forall c \in C_f, \forall s \in S(c), 0 \leq emptySeats < capacity(c)$ 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
- $\forall c \in C_f, \forall s \in S(c), endHour(s) > startHour(s)$
- $\forall c \in C_f, \forall s \in S(c), startHour(s) < pickupHour(\forall r \in R_f) < endHour(s) \wedge startHour(s) < destHour(\forall r \in R_f) \leq endHour(s)$

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.

Este problema é um caso do **Travelling Salesman Problem**.

Primeira Iteração

Floyd-Warshall bellmanFord Dijkstra A*