Universidade do Porto

Faculdade de Engenharia

FEUP

Projeto de Concepção e Análise de Algoritmos:

RideSharing: partilha de viagens (tema 1) - Parte I

Grupo D , turma 3:

Ana Rita Fonseca Santos	201605240
Filipe Carlos de Almeida Duarte da Cunha Nogueira	201604129
Pedro Maria Passos Ribeiro do Carmo Pereira	201708807

(2018/2019 - 2º Semestre)

Índice

1. Descrição do Tema
   1.1. Pré-cálculo
   1.2. 1ª iteração: Rota sem restrições temporais
   1.3. 2ª iteração: Rota com restrições temporais
2. Formalização do problema
   2.1. Dados de entrada
   2.2. Dados de saída
   2.3. Restrições
   2.4. Funções objetivo
3. Perspectiva da solução
4. Casos de utilização
5. Principais dificuldades encontradas no desenvolvimento do trabalho
6. Esforço dedicado por cada elemento do grupo
7. Bibliografia

1. Descrição do tema

Neste trabalho, pretende-se implementar uma aplicação para suportar o conceito de *ridesharing*. Esta aplicação irá conciliar as moradas, destinos e restrições temporais de diferentes utilizadores e gerar a melhor rota desde a origem do condutor até ao seu destino. Esta rota terá consideração os pontos de coleta dos outros passageiros e os seus respectivos destinos.
Os utilizadores estão espalhados pela cidade e a aplicação irá considerar outros critérios como familiaridade entre pessoas que partilham a mesma viagem, por exemplo.

Este problema pode ser decomposto em três iterações.

1.1 Pré-cálculo

Extração e processamento do grafo gerado pelo OpenStreetMap.

1.2. 1ª iteração: Rota sem restrições temporais

Numa primeira fase, vai se gerar um novo grafo com o povoamento da posição inicial e destinos de cada utilizador.
De seguida, calcula-se a distância mais curta entre pares de nós. Os nós representam cruzamentos e as ruas arestas. O peso de cada aresta do novo grafo vai ser o tempo que o condutor demora a chegar de um local (nó) a outro.

O objectivo nesta fase é calcular a rota mais curta que satisfaça as condições previamente referidas. A rota é calculada apenas considerando um veículo.

O utilizador que disponibiliza o veículo (condutor), vai passar por todos os utilizadores e leva-os até ao seu respectivo destino. A capacidade do veículo é previamente estipulada e o condutor, sempre que atingir a sua capacidade máxima, não recolhe mais nenhum utilizador e apenas prioriza os pontos de entrega. Assim que a capacidade do veículo for diminuída, o condutor já pode considerar recolher outros utilizadores.

Os utilizadores fora do veículo irão ser caracterizados em função dos seus destinos, da proximidade das suas origens, e das restrições da viagem que pretendem realizar considerando a posição do condutor. Nesta fase ainda não são consideradas restrições temporais. Desta forma é composta uma lista de prioridades com os seguintes critérios: destino, origem e a familiaridade entre pessoas que partilham a mesma viagem. Sempre que a posição do condutor muda, a lista de prioridade é atualizada de acordo com as restrições apresentadas anteriormente.

A rota começa no ponto de partida do utilizador com veículo.

Este inicialmente parte em busca de um utilizador com uma maior prioridade.
O condutor irá recolhendo utilizadores até a capacidade máxima do veículo seja atingida. Caso exista um destino dos utilizadores que estão dentro do veículo que seja mais próximo do que a origem do utilizador com maior prioridade, o condutor dirige-se para o destino.

Conforme o condutor for deixando os utilizadores nos seus pontos de destino, estes já não vão ser mais contabilizados. O condutor acaba a sua rota quando chega ao seu próprio destino podendo este ser também um destino comum de outro passageiro.

1.3. 2ª iteração: Rota com restrições temporais

Nesta iteração, irá ser feita uma análise de compatibilidade. De acordo com a lista de prioridades atingida na iteração anterior, a aplicação irá agora compatibilizar com as restrições de todos os utilizadores dentro do veículo.

Caso o tempo restante mais a tolerância de qualquer passageiro dentro do veículo não seja suficiente para o condutor ir um outro utilizador, o condutor dá prioridade ao passageiro.
Esta restrição de tempo é também tendo em conta a restrição de tempo do utilizador com maior prioridade. Ou seja, caso as restrições de tempo deste último não sejam compatíveis com as restrições de tempo do passageiros, é agora considerado o utilizador seguinte da lista de prioridades. Se nenhum utilizador da lista de prioridade for compatível, então o condutor vai optar por um destino.

2. Formalização do problema

2.1. Dados de entrada

Grafo G = (Vi, Ei) dirigido pesado, que consiste num conjunto de:

- vértices Vi, moradas válidas para utilizadores, cruzamentos em que os mesmos podem se encontrar e ser apanhados pelo condutor.
- arestas Ei, estrada entre duas moradas em que o peso respectivo de cada aresta
  representa o tempo que o veículo demora a atravessar a estrada.

Lista de utilizadores que populam o grafo.
Cada utilizador Ui é caracterizado por:

- Si, morada de partida (Si $\in$ Vi)
- Di, morada de destino (Di $\in$ Vi)
- TolS, tempo de tolerância de partida
- TolD, tempo de tolerância de chegada
- TSi, tempo pretendido de partida

- TDi tempo pretendido de chegada

Condutor C (derivado de utilizador), representando o utilizador cujo percurso se quer otimizar, composto por:

- Atributos de utilizador
- N, capacidade máxima do veículo
- currPass, lista passageiros num dado momento
- totalPass, lista de passageiros total, (todos os transportados)

2.2. Dados de saída

Grafo G = (Vf, Ef) dirigido pesado, em que Vf $\subset$ Vi e Ef $\subset$ Ei, representando apenas os nós e arestas do percurso otimizado para o condutor.

Condutor C: estado final do utilizador condutor, em que a lista temporária de passageiros deverá estar vazia, e a lista permanente populada com os utilizadores transportados ao longo do percurso.

2.3. Restrições

N >= 0, se o condutor não tivesse lugares disponíveis, não seria necessário otimizar o seu percurso que seria direto.

Size(Condutor.currPass) <= N, o condutor não poderá transportar mais passageiros que o máximo de lugares disponíveis no carro.

Restrições temporais de cada passageiro e do condutor. Condutor deve chegar à posição final dentro do limite de tempo, e deve-se assegurar que consegue transportar cada passageiro de acordo com as suas próprias restrições temporais.

2.4. Funções objetivo

A solução óptima para este problema é atingida quando a lista de prioridades referida anteriormente é construída de forma a que o condutor escolha sempre ou o utilizador com maior prioridade ou o destino mais próximo ou dê prioridade a um passageiro de acordo com as suas restrições temporais. Para tal é necessário maximizar a função **f** a seguir descrita na perspectiva de solução:

- f = Σ(wi*Ci)

Após uma boa implementação desta função, é possível comparar os utilizadores fora do carro com os passageiros do veículo.

Para atingir o potencial máximo de otimização do problema é necessário também que a função h, que representa o número de passageiros recolhidos, seja maximizada e, simultaneamente, a função g, que representa o tempo total de viagem do condutor (onde t(e) representa o tempo de viagem de cada aresta), seja minimizada.

- h = |Condutor.totalPass|

- $g = \sum_{e \in E} t(e)$

Desta forma é encontrada a melhor solução que consiste em respeitar todas as restrições do problema enquanto se recolhe o maior número de utilizadores.

3. Perspectiva da solução

A solução deste problema terá como objectivo maximizar o número de utilizadores que o condutor entrega nos seus respectivos destinos, de acordo com as restrições de todos os utilizadores envolvidos. Para atingir esta solução é necessário implementar certos algoritmos:

- No cálculo da lista de prioridade referida anteriormente, irá ser utilizado o seguinte algoritmo:
    - f = Σ(wi*Ci)
      f - valor unitário de prioridade do respectivo utilizador.
      Ci - critério a considerar.
      wi - peso do critério i.

- No cálculo do peso de cada aresta do novo grafo que foi gerado a partir do grafo inicial, podem ser utilizados 2 algoritmos.
    - Caso o grafo seja esparso, ou seja, o número de arestas |E| seja próximo do número de vértices |V|, o algoritmo a ser utilizado será o de Dijkstra que apresenta uma complexidade de O((|V|+|E|) * log |V|).
    - Caso o grafo seja denso, ou seja, o número de arestas |E| seja maior que o quadrado do número de vértices $|V|^2$, o algoritmo a ser utilizado será o de Floyd-Warshall que apresenta uma complexidade de O(|V|^3).

  A aplicação de diferentes algoritmos para a mesma situação (calculo do peso das arestas), é relevante na otimização do tempo de execução. Apresentado estas duas opções, garante-se que a solução do problema é mais versátil.

4. Casos de utilização
Operações CRUD (*Create, Read, Update and Delete*) aos utilizadores que se propagarão como modificações no grafo a ser trabalhado.

Botão para inicializar a solução especificada neste trabalho.

5. Principais dificuldades encontradas no desenvolvimento do trabalho
Aquando da escrita deste relatório preliminar, a principal dificuldade encontrada foi a definição e compreensão dos requisitos do trabalho proposto.

6. Esforço dedicado por cada elemento do grupo
A escrita deste relatório preliminar foi feita em conjunto, com todos os elementos do grupo presentes e contribuindo para um objetivo em comum, creemos que o trabalho e esforço de cada membro foi igual por esta razão.

Ana Rita Santos	-	33.3%
Filipe Carlos Nogueira	-	33.3%
Pedro Pereira	-	33.3%

7. Bibliografia
Material de estudo de Teoria de Grafos disponibilizado na página da UC de CAL e utilizado na realização deste relatório preliminar.

https://moodle.up.pt/pluginfile.php/227078/mod_label/intro/06.grafos1.pdf

https://moodle.up.pt/pluginfile.php/229148/mod_label/intro/07.grafos2.pdf

https://moodle.up.pt/pluginfile.php/231896/mod_label/intro/08.grafos3.pdf

https://moodle.up.pt/pluginfile.php/231896/mod_label/intro/09.grafos4.pdf

https://moodle.up.pt/pluginfile.php/237688/mod_label/intro/13.grafos8.pdf

https://moodle.up.pt/pluginfile.php/237688/mod_label/intro/14.grafos9.pdf