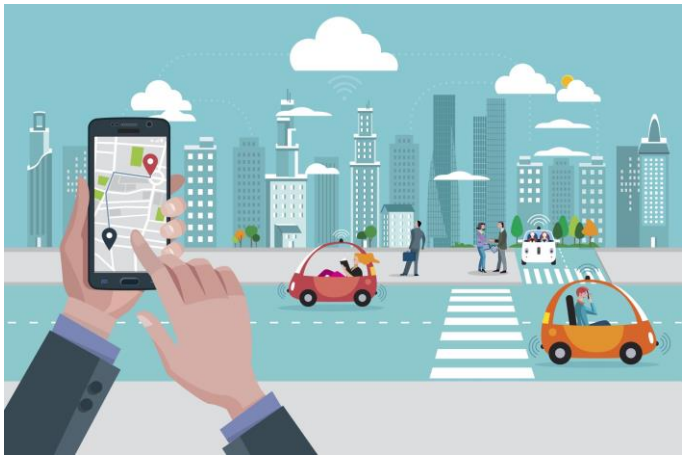
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO



Tema 1
MeetupRider: partilha de viagens em grupos

MIEIC

Conceção e Análise de Algoritmos

Turma 2 - Grupo 2

24/04/2020

Clara Gadelho	up201806309@fe.up.pt
Diogo Nunes	up201808546@fe.up.pt
João Fernandes	up201806724@fe.up.pt

Índice

1. Descrição do tema

Uma grande parte da população realiza as suas viagens entre casa e um dado destino de carro. Contudo, normalmente, realizam-no cada um no seu veículo, não contribuindo para a preservação do meio ambiente e acabando por influenciar o trânsito nas estradas.

Para que isso não aconteça, iremos adotar o conceito de ridesharing (partilha de viagens), com o intuito de evitar que os condutores viajam cada um no seu veículo, e maximizar a capacidade de cada carro.

1.1. 1ª Iteração

Menor percurso para passageiros com o mesmo destino do condutor independentemente do seu ponto de partida e sem considerar restrições horárias

Numa primeira fase, a estratégia consiste em considerar que cada carro transporta somente passageiros com o mesmo destino do condutor, para garantir que os pontos de passagem intermédios são somente os pontos de partida dos passageiros e não também os seus destinos. Para além disso não são tidas em conta horas específicas de partida nem chegada.

Assim, o carro parte do ponto de partida do condutor, e vai dirigindo-se aos pontos de partida dos passageiros por ordem de proximidade, sendo sempre usado o caminho mais curto, até todos os lugares vagos no carro estarem ocupados ou já não houver mais pessoas com o mesmo destino ou para as quais não exista um caminho possível. Quando uma destas condições se verifica, o carro dirige-se para o ponto de destino e a viagem é concluída.

1.2. 2ª Iteração

Menor percurso para passageiros com o mesmo destino do condutor independentemente do seu ponto de partida considerando restrições horárias

Numa segunda abordagem, acrescenta-se à estratégia anterior as restrições horárias impostas pelos passageiros e condutor, sendo que tem que ser assegurado que todas as pessoas transportadas partem do seu ponto inicial depois da hora mínima estipulada e chegam destino antes da hora limite.

1.3. 3ª Iteração

Menor percurso para passageiros com partida e/ou destino diferentes do condutor considerando as restrições horárias

Por fim, é acrescentada a possibilidade de os passageiros terem destinos diferentes dos do condutor. Esta é por isso a situação mais complexa, dado que para cada passageiro é necessário não só acrescentar o seu ponto de partida ao percurso, mas também o seu destino, e calcular a ordem mais vantajosa de incluir esses destinos no percurso, tendo em conta não só as distâncias, mas também as restrições horárias de cada pessoa.

2. Formalização do problema

2.1. Dados de entrada

• V: Pontos – conjunto de pontos que representam locais num dado mapa. Cada ponto é caracterizado por:

- idPonto: identificador inteiro único para cada ponto;
- morada: morada completa correspondente ao ponto;
- coordenadas: Coordenadas geográficas do ponto no mapa;
- adj: conjunto de arestas adjacentes;

• E: Estradas – conjunto das estradas que ligam os vários locais no mapa. Cada estrada é caracterizada por:

- idEstrada: identificador inteiro único para cada estrada;
- origem: identificador do ponto de origem da estrada;
- destino: identificador do ponto de destino da estrada;
- distancia: comprimento da estrada;
- tempo: tempo médio para percorrer a estrada;

• G(V, E) – grafo dirigido pesado, sem pesos negativos, cujos vértices V representam os vários pontos do mapa e as arestas E representam as estradas que ligam os diversos vértices.

• Pessoas - conjunto de pessoas que precisam ou dão boleia. Cada utilizador é caracterizado por:

- idUtilizador: identificador inteiro único para cada pessoa da rede;
- origem: identificador do ponto de partida do utilizador;
- destino: identificador do ponto onde o utilizador pretende chegar;
- horaPartida: hora a partir da qual o utilizador pode começar a sua viagem;
- horaChegada: hora máxima para o utilizador chegar ao seu destino;
- condutor: se possui ou não carro próprio;

• Carros - conjunto de carros dos utilizadores que podem dar boleias. Cada carro é caracterizado por:

- idCarro: identificador inteiro único de cada carro;
- condutor: identificador do utilizador que possui o carro;
- numPassageiros: número de passageiros atual
- capacidade: número de passageiros que o carro pode levar;

2.2. Dados de saída

- Passageiros: conjunto de utilizadores aos quais cada carro dará boleia;
- Caminho: sequência ordenada de pontos (que pertencem a V) que devem ser percorridos, começando no ponto de partida do condutor, passando por todas as origens e destinos das pessoas que recebem a boleia desse condutor (U) até ao destino deste.

2.3. Restrições

∀ v1, v2 ∈ Pontos: v1.idPonto ≠ v2.idPonto ∧ v1.morada ≠ v2.morada ∧ v1.coordenadas ≠ v2.coordenadas

∀ e1, e2 ∈ Estradas: e1 ≠ e2 ⇒ e1.idEstrada ≠ e2.idEstrada

∀ e ∈ Estradas: e.distancia > 0 ∧ e.tempo > 0 ∧ e.origem ≠ e.destino

∀ u1, u2 ∈ Pessoas: u1 ≠ u2 ⇒ u1.idUtilizador ≠ u2.idUtilizador

∀ u ∈ Pessoas: u.horaPartida < u.horaChegada ∧ u.origem ≠ u.destino

∀ c1, c2 ∈ Carros: c1 ≠ c2 ⇒ c1.idCarro ≠ c2.idCarro

∀ c ∈ Carros: c.numPassageiros >= 0 ∧ c.capacidade > 0

∀ c ∈ Carros: c.numPassageiros <= c.capacidade

∀ p ∈ Passageiros: p ∈ Pessoas

∀ v ∈ Caminho: v ∈ Pontos

2.4. Função objetivo

O objetivo principal passa por minimizar o tempo de viagem de cada pessoa, enquanto minimizamos o número de carros utilizados e que todos cheguem aos seus destinos até à hora de chegada que cada pessoa disponibilizou.

$\forall\ p \in$ Pessoas :

p.horaPartida + $\sum_{k=0}^{n-2}$ Estrada (Ponto k, Ponto k+1).tempo <= p.horaChegada

(n = tamanho do caminho de cada pessoa ao seu destino)

3. Perspetiva de solução

Inicialmente teremos que implementar algum tipo de redução do grafo, considerando para isso apenas o ponto de partida do condutor e os nós que correspondem ao ponto de partida dos diversos clientes. Para tal iremos usar ou Dijkstra para cada um desses pontos ou Floyd-Warshall para todos os pontos e de tal forma eliminamos os nós e arestas desnecessários, sobrando então as distâncias mínimas entre os nós essenciais.

Antes de começar as iterações propriamente ditas irá ser testado o caso ideal de os passageiros estarem já no trajeto do condutor e de terem destinos iguais, não sendo necessário qualquer desvio para o condutor. Este caso é um caso à parte e caso seja verificado reduz de forma bastante significativa o esforço computacional tal como o tempo necessário para o condutor e passageiros chegarem ao seu destino. Como é um caso à parte não foi considerado como uma iteração, mas sim como uma otimização inicial.

Após isso teremos a distância euclidiana do ponto de partida do condutor ao seu destino, como na primeira iteração não consideramos tempos limite vamos usar um algoritmo ganancioso que, a partir do nó em que o condutor se encontra no momento atual calcula o nó mais próximo de um passageiro, tendo em conta que estes devem ter todos o mesmo destino. Esse algoritmo é repetido até o limite de passageiros do carro ser atingido ou até não serem encontrados mais nenhum passageiro com esse mesmo destino em comum.

Numa 2ª iteração iremos usar uma função que calcule o tempo médio de deslocação para cada aresta para podermos ter em conta os tempos limite tanto para o condutor como para os clientes. Usaremos um algoritmo ganancioso semelhante ao anterior exceto que, ao encontrar um novo cliente com o mesmo destino é calculado o tempo de desvio, ou seja, o tempo que iria demorar ao condutor a ir até à localização do cliente somando-o com o tempo que demora a percorrer desde a localização do cliente até ao destino final anteriormente combinado. Caso ao fazer esse desvio o tempo não seja superior ao limite de tempo de nenhuma pessoa (condutor ou passageiro), o nó de partida do cliente é adicionado ao percurso do carro e é repetido esse raciocínio para adicionar clientes até que o carro esteja cheio ou não seja possível acrescentar mais nenhum cliente sem provocar atrasos aos restantes passageiros já a bordo.

Finalmente numa última iteração iremos considerar destinos diferentes entre clientes e passageiros. Neste passo temos de considerar que à medida que são adicionados passageiros ao carro os seus destinos têm prioridade ao destino do condutor, isto pois o destino do condutor será o nó final na fila de prioridade que determina o fim da viagem. Para tal temos de acrescentar mais um tempo ao algoritmo ganancioso anterior, que corresponde ao tempo do destino do passageiro ao destino do condutor. À medida que os passageiros são adicionados aumenta a complexidade da solução porque têm de ser sempre consideradas as compatibilidades de tempo com os passageiros que já foram recolhidos tais como as distâncias entre os vários destinos.

4. Casos de utilização

A solução a implementar mostrar-se-ia bastante útil ao ser integrada numa aplicação móvel de partilha de boleias, na qual os utilizadores pudessem registar-se para dar ou receber boleias.

Para isso, seriam necessárias as seguintes funcionalidades:

- Utilizar a informação proveniente de um mapa;
- Guardar a informação dos utilizadores, bem como a dos respetivos carros;
- Permitir a adição de novos utilizadores e carros;
- Calcular os percursos;
- Visualizar a informação dos percursos;
- Visualizar a informação dos utilizadores.

5. Conclusão

As principais dificuldades que encontramos ao realizar este relatório passaram pela definição das iterações mais rentáveis a fazer, de modo a aproveitar cada uma delas com o que devia ser feito e não realizar iterações a mais que não fossem extremamente necessárias. Também tivemos um impasse na formalização do problema, tendo em conta a complexidade do tema. Devido a ainda não termos começado a implementar código, também não temos a certeza dos algoritmos a usar, apesar de termos noção de quais poderemos usar. Além disso, não temos forma de testar como vão funcionar com dados na escala de um mapa real, com muitos mais dados de entrada quando comparados com os que usamos nas aulas práticas.

O esforço realizado foi dividido por todos os elementos do grupo, tendo discutido todos os pontos do relatório em conjunto.

6. Bibliografia

- Slides das aulas teóricas, R. Rossetti, L. Ferreira, H. L. Cardoso, F. Andrade, FEUP, MIEIC, CAL
- “Introduction to Algorithms”, Second Edition, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, The MIT Press, 2001;
- Solving Large Carpooling Problems Using Graph Theoretic Tools, https://www.auhl.be/Documents/datasim/Summer%20School%202014/(L.3A)%20PPT-Irith-Hartman.pdf;