Pesquisa Operacional para o Desenvolvimento
Vol. 12, e12001, p. 1-11, 2020

ISSN 1984-3534

TUTORIAL CONVIDADO
INSTÂNCIAS PARA ROTEAMENTO DE VEÍCULOS USANDO DADOS ABERTOS[1]

Carlo S. Sartori[a], Luciana S. Buriol[a]*

[a] Instituto de Informática, Departamento de Informática Teórica
Universidade Federal do Rio Grande do Sul, Porto Alegre - RS, Brasil

Recebido 23/02/2020, aceito 06/05/2020

RESUMO

Este tutorial apresenta um método para geração de instâncias de Problemas de Roteamento de Veículos com base em dados abertos. O objetivo principal é obter um processo automatizado e replicável que permita a geração de instâncias com características próximas da realidade. O trabalho descreve um processo de obtenção de coordenadas geográficas dentro de uma área de interesse, bem como ferramentas a serem utilizadas para o cálculo de distâncias e tempos de viagem respeitando a malha urbana local. Além disso, apresenta métodos para a geração de características específicas do problema de roteamento para completar o processo de geração de uma instância. Espera-se que este tutorial mostre como o uso de dados abertos pode ser usado para gerar instâncias que permitam novas análises de algoritmos de roteamento e o impacto que certas características das instâncias têm neles e nas soluções encontradas.

Palavras-chave: Problema de roteamento de veículos, Instância, Dados abertos.

ABSTRACT

This tutorial presents a method to generate instances for Vehicle Routing Problems using open access data. The goal is to provide an automatic and reproducible technique which allows the generation of instances with certain characteristics from real-world applications. The work describes how to obtain geographic coordinates within an area of interest, as well as tools for the computation of distance and travel times in the local urban grid. Furthermore, it presents methods on how to generate other specific characteristcs of vehicle routing problems to fully generate an instance. We hope this tutorial is able to show how the use of open access data can be used to generate instances that create new opportunities to analyse algorithms and solutions for routing problems and the impact that certain instance characteristics may have on them.

Keywords: Vehicle routing problem, Instance, Open data.

* Autor para correspondência. E-mail: buriol@inf.ufrgs.br
DOI: 10.4322/PODes.2020.001

[1] *Todos os autores assumem a responsabilidade pelo conteúdo do artigo.*

1. Introdução

O Problema de Roteamento de Veículos (PRV) é um tópico da Pesquisa Operacional (PO) com diversas aplicações práticas em logística e transporte, e que tem sido estudado há décadas (Dantzig e Ramser, 1959). O objetivo do problema é construir rotas de veículos que atendam requisições de clientes com o menor custo operacional possível (e.g., combustível, quilometragem, ou número de veículos usados). Além disso, as rotas devem respeitar restrições adicionais do problema, como capacidade de carga dos veículos, tempo de operação nos clientes, precedências entre visitas, e diversas outras. De fato, restrições adicionais definem PRVs diferentes modelando cenários reais de logística.

A literatura do PRV é vasta e sua revisão não faz parte do escopo deste tutorial. Para este fim, aconselhamos aos interessados a leitura dos trabalhos de Laporte (2009) e Toth e Vigo (2014) para uma visão abrangente a respeito de tópicos do problema.

No desenvolvimento de métodos de solução para problemas da PO, é comum a literatura possuir conjuntos de instâncias padrões (*benchmarks*) que permitem comparar técnicas diferentes. Os PRVs possuem conjuntos bem conhecidos, como o do PRV com Capacidades (Uchoa et al., 2017), do PRV com Janelas de Tempo (Solomon, 1987), e do Problema de Coleta e Entrega com Janelas de Tempo (Li e Lim, 2003). Todos estes conjuntos possuem instâncias definidas em um plano cartesiano 2D, onde pontos gerados aleatoriamente ou de acordo com alguma distribuição definem clientes. As distâncias são calculadas de forma linear (e.g., distância Euclidiana).

Por um lado, gerar instâncias desta forma é simples, rápido e facilmente replicável. Com poucas linhas em um artigo é possível descrever todo o processo. Por outro lado, tais instâncias não necessariamente simulam características de cenários de transporte reais. Ao mesmo tempo, obter dados reais é muitas vezes difícil e nem sempre supre a necessidade do pesquisador (e.g., instâncias pequenas), ou não podem ser publicados devido a barreiras legais.

Este tutorial busca descrever um método para contornar estes problemas e gerar instâncias do PRV utilizando dados abertos sobre endereços e coordenadas reais, juntamente com o cálculo de distâncias e tempos de viagem considerando uma malha urbana completa, também proveniente de dados abertos. O processo de geração é replicável e pode ser automatizado para gerar instâncias do PRV em qualquer área de interesse para a qual os dados necessários estejam disponíveis em repositórios públicos.

2. Geração de Instâncias com Dados Abertos

A geração de instâncias pode ser dividida em três partes principais: (i) a escolha das coordenadas geográficas para aproximar a distribuição de clientes na região de interesse, (ii) o cálculo da matriz de distâncias ou tempos de viagem, e (iii) a geração de características específicas que dependem da variação de PRV que a instância representará. Descrevemos cada etapa nas subseções a seguir, com um enfoque especial para as duas primeiras, onde o uso de dados abertos consegue obter o maior impacto.

2.1. Distribuições Populacionais e Coordenadas Geográficas

Na geração das instâncias, queremos obter coordenadas geográficas que façam parte de uma determinada região (e.g., cidade, estado, ou país). Além de localizarem clientes dentro de uma área, elas servem para realizar o cálculo de distâncias e tempos de viagem que respeitam a malha urbana local. Uma possibilidade seria gerar coordenadas (latitude e longitude) aleatoriamente dentro de uma cidade, porém isto não é muito diferente de gerar pontos no plano, além de trazer problemas como a geração de locais dentro d'água ou em lugares impróprios.

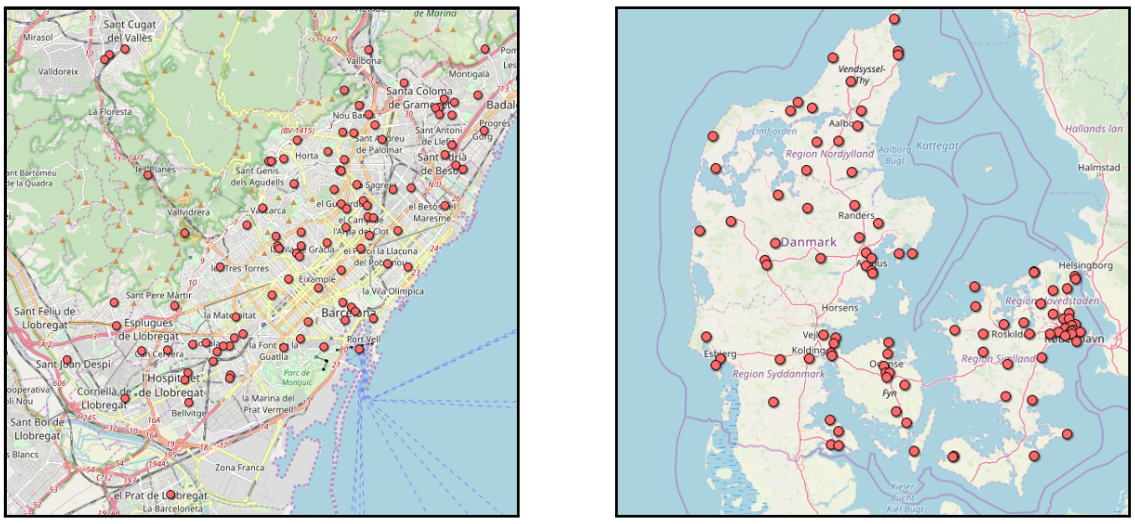
Para contornar estas dificuldades, sugerimos o uso de base de dados abertas contendo endereço para diferentes partes do mundo. O projeto OpenAddresses (2017) mantém endereços completos, incluindo coordenadas geográficas, para diversas regiões. Neste tutorial, vamos usar

algumas áreas como exemplo. Na cidade de Barcelona, na Espanha, existem 129.460 coordenadas cadastradas, já para toda a Dinamarca são 2.512.925. Em Berlim, na Alemanha, são 371.265, e na cidade de Rio de Janeiro (capital) 728.622. São conjuntos de coordenadas suficientes para gerar instâncias de grande escala para PRVs.

A maior vantagem de utilizarmos estas coordenadas é a de que distribuições populacionais em cidades raramente são simétricas. Algumas regiões são mais populosas que outras devido a fatores como terreno ou proximidade aos grandes centros. Inclusive, algumas das fontes do OpenAddresses (2017) são dados disponibilizados por governos ou instituições que realizaram censos a respeito da distribuição populacional. Por exemplo, muitos dados brasileiros na plataforma foram obtidos do censo do Instituto Brasileiro de Geografia e Estatística (2010). Portanto, estes dados abertos nos proporcionam, de forma indireta, uma distribuição na região de interesse, já que áreas mais populosas potencialmente possuem mais endereços. Se simplesmente escolhermos aleatoriamente locais dentre os cadastrados nos conjuntos de dados, obtemos uma distribuição não uniforme para a seleção nessa região.

Para ilustrar a argumentação anterior, apresentamos as Figuras 1(a) e 1(b), referentes a seleções de locais na cidade de Barcelona e para toda a Dinamarca, respectivamente. Ambas possuem 100 locais escolhidos aleatoriamente dentre os conjuntos de coordenadas citados. Todas as figuras foram geradas usando a ferramenta *GPS Visualizer* criada e disponibilizada por Schneider (2003). Na Figura 1(a), notamos que a maior parte dos pontos estão nas regiões urbanizadas da cidade, isto é, seguindo a distribuição subjacente da população, ainda que alguns poucos locais tenham sido escolhidos fora das principais regiões. Para a Figura 1(b), percebemos que há a formação de um agrupamento de pontos (*cluster*) nas proximidades da capital, Copenhague, devido à maior concentração urbana nesta região e ao número de endereços cadastrados na base de dados.

Figura 1: Exemplo de seleção aleatória de 100 coordenadas em duas regiões distintas. Círculos vermelhos são pontos selecionados.



(a) Barcelona (b) Dinamarca

Fonte: Elaborado pelos autores.

É claro que desta forma, a distribuição de coordenadas está intimamente ligada aos dados de entrada do gerador de instâncias. É esperado que se os dados possuem algum *viés*, este acabará sendo reproduzido na geração da instância. Para esclarecer esta afirmação, vamos utilizar outra base de dados aberta organizada por Donovan e Work (2016), a qual mantém dados a respeito de corridas de táxi na cidade de Nova Iorque ao longo dos anos de 2010 a 2013. Este conjunto contém milhões de cadastros, cada um com origem e destino da corrida (útil para problemas de coleta e

entrega, ou *dial-a-ride*), bem como horário, duração, e número de passageiros. Ao selecionar 100 locais aleatórios deste conjunto, entre origens e destinos, notamos algo peculiar, apresentado na Figura 2. Uma enorme parcela das coordenadas formaram um grande *cluster* na ilha de Manhattan, com poucos pontos fora. A razão, é claro, se deve ao fato de que Manhattan não só é uma região densamente populosa, mas também repleta de turistas e atividades comerciais, com muitas pessoas fazendo uso de táxis para se locomover no trânsito da ilha. Assim, este conjunto de dados, ainda que bastante completo, reflete um viés da realidade na geração das instâncias. Por outro lado, é uma característica que pode ser útil para certos problemas e trazer análises interessantes, já que as distâncias e tempos de viagens neste caso são majoritariamente pequenos.

Figura 2: Exemplo de seleção aleatória de 100 coordenadas na cidade de Nova Iorque.

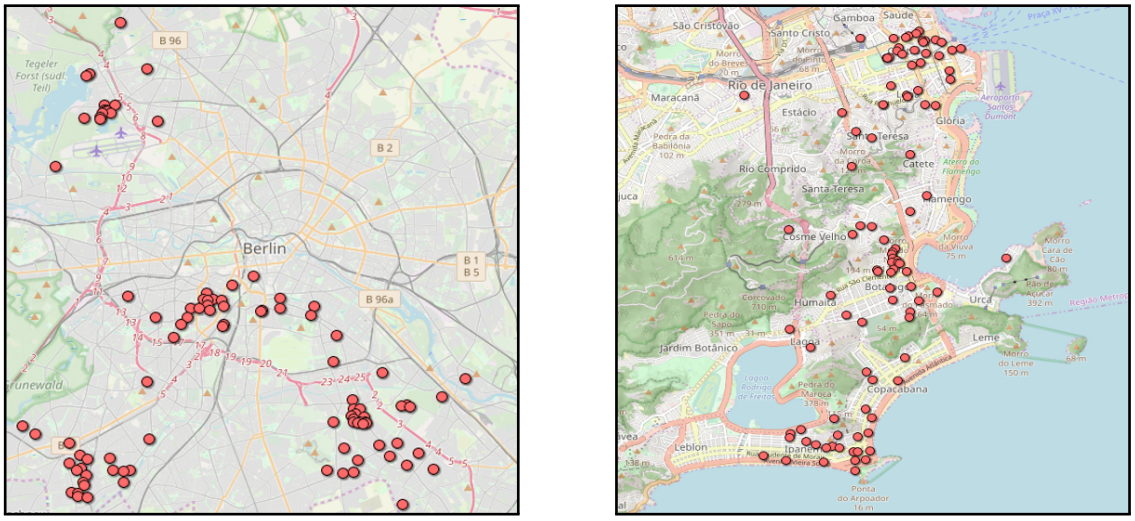
Fonte: Elaborado pelos autores.

Além da forma básica de seleção aleatória, é possível unir a isto uma distribuição diferente, como a de agrupamentos, para forçar a geração de grupos dentro de uma região. As Figuras 3(a) e 3(b) apresentam uma instância com 100 coordenadas em Berlim e no Rio de Janeiro, respectivamente. Para a primeira, quatro *clusters* foram gerados, enquanto que para a segunda foram três. Berlim é uma cidade bastante urbanizada e quase aproxima um plano 2D, portanto a geração por agrupamento se parece muito com o que ocorreria em uma instância no plano comum. Por outro lado, no Rio de Janeiro, com seus acidentes geográficos, notamos a geração de agrupamentos não só respeitando estes obstáculos, mas também seguindo áreas de maior movimentação da cidade.

A geração dos *clusters* neste exemplo seguiu a proposta de Uchoa et al. (2017). Primeiro, escolhemos um conjunto S de locais para servirem de sementes dos agrupamentos (uma semente por agrupamento). Em seguida, para cada coordenada x no conjunto de entrada, calculamos a probabilidade $p(x)$ de x pertencer à instância usando a distribuição $p(x) = \sum_{s \in S} \exp(-D(s,x)\delta)$, onde a função $D(s,x)$ computa uma métrica de distância entre s e x e δ é um parâmetro de densidade, indicando quão denso os agrupamentos devem ser. A função D pode ser uma dentre várias, como Euclidiana, *Haversine*, ou mesmo uma computação sobre a malha urbana como descrito na Seção 2.2. Neste exemplo, utilizamos a distância *Haversine*.

É claro que, tratando-se de dados abertos, nem tudo é perfeito. Para o uso dos endereços, precisamos obter os arquivos texto (formato separado por vírgulas, ou CSV) e em muitos casos filtrá-los antes. Por exemplo, para obter as coordenadas de Barcelona, foi preciso filtrar um arquivo contendo coordenadas de uma região muito maior na Espanha, selecionando todas as coordenadas que pertenciam à região de Barcelona definida por uma secção retangular ao redor da cidade. A filtragem também pode servir para mudar o enfoque em uma área densa, como o Rio de Janeiro, onde filtramos coordenadas da região central da cidade, já que vários endereços também estavam

Figura 3: Exemplo de seleção de 100 coordenadas aleatórias em duas regiões distintas.



(a) Berlim (b) Rio de Janeiro

Fonte: Elaborado pelos autores.

cadastrados em bairros mais distantes. Assim, é necessário algum trabalho na obtenção e filtragem dos dados, o que mesmo assim pode ser automatizado com *scripts* de maneira simples.

2.2. Distâncias e Tempos de Viagem

Uma vez que obtemos as localizações de clientes dentro da área de interesse, podemos calcular a matriz de distâncias ou tempos de viagem entre estes pontos, a qual é necessária para definir uma instância do PRV. Uma forma de calcular é usar distâncias lineares. Porém, uma das razões para utilizarmos coordenadas reais foi a possibilidade de calcular distâncias e tempos sobre uma malha urbana real. Portanto, vamos descrever apenas a última forma.

Como parte deste tutorial, sugerimos o uso da ferramenta de código aberto Open Source Routing Machine (OSRM) proposto por Luxen e Vetter (2011). O OSRM implementa algoritmos do estado-da-arte para o cálculo de distâncias e tempos de viagem mais curtos entre origem e destino. O programa é capaz de calcular estas informações sobre mapas do projeto OpenStreetMap (2017) (OSM), o qual também é aberto. Uma grande vantagem de utilizar o OSRM é a velocidade de resposta, já que todo o processo pode ser executado na máquina local e, portanto, a velocidade depende unicamente da capacidade do computador que executa a operação.

Antes de iniciar o uso do OSRM, é preciso obter os dados dos mapas que iremos usar, isto é, mapas do OSM. Para tal, podemos obter o mapa mundial, ou de regiões menores, diretamente na página do projeto OpenStreetMap (2017), ou no repositório da Geofabrik (2018).

O código e as instruções de instalação e uso para o OSRM estão disponíveis no repositório do projeto OSRM-backend (2020). O programa pode ser usado como aplicação externa fazendo requisições HTTP locais ao OSRM e recebendo a resposta, ou como uma parte integrada do código do gerador de instâncias, se este for implementado em linguagem `C++14` ou fizer uso de bibliotecas dinâmicas. Em qualquer um dos casos, o OSRM permite o cálculo de uma tabela ao fornecermos uma sequência de coordenadas, efetivamente computando a matriz e informando para cada entrada (i, j) da tabela, a distância e o tempo de viagem entre as coordenadas i e j. A resposta é dada em padrão JSON, e acompanha, além destas informações básicas, uma série de outros dados a respeito dos caminhos que podem ser utilizados.

Porém, para realizar as requisições e obter respostas, é preciso pré-processar os mapas OSM e criar arquivos de padrão OSRM que são usados para o cálculo dos caminhos de forma mais

eficiente. O pré-processamento pode ser demorado e consumir bastante memória na execução, dependendo do tamanho do mapa. Por exemplo, para todo o Brasil, o OSRM exigiu cerca de 8 GB de memória RAM e 30 minutos de execução em um processador Intel i7 930 com 2.8 GHz. No pré-processamento dos mapas, o usuário pode definir as características para o cálculo dos caminhos, como função objetivo e tipo de veículo. Este conjunto de características é chamado *perfil* e é um arquivo texto (em linguagem LUA) contendo instruções para o OSRM. Para cada perfil diferente, é necessário realizar um novo pré-processamento e gerar arquivos OSRM novos. Abaixo descrevemos algumas das principais configurações.

O OSRM pode usar três objetivos diferentes no cálculo dos caminhos: (1) menor distância (denotado no perfil por `distance`), (2) menor tempo (`time`), ou (3) menor tempo favorecendo certas rotas (`routability`). O objetivo (1) informa a rota de menor distância, e o (2) a de menor tempo. Já o (3) calcula a rota de menor tempo, mas com preferência para certas rotas de maior acessibilidade. Um fator importante na escolha do cálculo dos caminhos diz respeito à *desigualdade triangular*. Se utilizarmos o (1) para o cálculo, a matriz de distâncias garantidamente respeita a desigualdade, por outro lado, a matriz de tempos não tem garantia nenhuma de respeitar a desigualdade (note, um caminho mais curto pode ser *mais demorado*). Da mesma forma, se utilizarmos o (2), a matriz de tempos respeita a desigualdade triangular, mas a matriz de distâncias não necessariamente respeita. Já para o (3) nenhuma das duas matrizes tem garantia de respeitar a desigualdade, pois o fator acessibilidade pode indicar um caminho um pouco mais longo, mas de melhor acesso. Logo, é preciso analisar o problema em questão e verificar que tipo de matriz é necessária (e.g., grande parte das técnicas para PRV com Janelas de Tempo exige a desigualdade triangular nos tempos).

Além disso, diferentes modos de transporte podem ser utilizados no cálculo dos caminhos pelo OSRM. A ferramenta consegue usar modos como *a pé* (`mode.walking`), *de bicicleta* (`mode.cycling`), e *de veículo motorizado* (`mode.driving`). A diferença dos meios de transporte não está só na velocidade, mas também no acesso a certos caminhos (e.g., pedestres podem acessar áreas onde carros não são permitidos). Ainda, podemos definir características do transporte que interferem no cálculo, como velocidade máxima, peso e tamanho. O primeiro, é claro, limita a velocidade máxima do transporte (e.g., um caminhão que possui velocidade máxima legal). Já os dois últimos, restringem veículos grandes a não percorrerem ruas pequenas. Esta diferença de perfis dos veículos pode ser útil, por exemplo, na geração de instâncias com veículos heterogêneos. O repositório do projeto já disponibiliza alguns perfis de exemplo. A Figura 4 apresenta um trecho do perfil base para *carros* (o modo usual de transporte em PRVs) a fim de ilustrar essa discussão.

Para este perfil, a função objetivo (`weight_name`) é a terceira opção descrita anteriormente (menor tempo com preferência por acessibilidade). Para o cálculo da duração do trajeto, algumas penalidades são informadas, incluindo pesos para desvios em formato de U (`u_turn_penalty`), para semáforos na via (`traffic_light_penalty`), e para curvas quaisquer (`turn_penalty`). Os pesos são informados em segundos e efetivamente adicionados ao tempo total do trajeto que contenha alguma destas características. No caso particular da `turn_penalty`, o valor informado é o maior incremento possível da duração do trajeto (neste caso 7,5 segundos), e o valor exato a ser adicionado em cada curva depende do ângulo desta curva. Não existem limites para as penalidades, e um valor alto para uma dada penalidade não significa que caminhos com ela serão necessariamente evitados. É claro que estes valores só fazem sentido para as funções objetivo `time` e `routability`, já que a `distance` não leva em conta o tempo de deslocamento. O perfil também define o modo padrão de deslocamento (`default_mode`), que neste caso é dirigindo. Para o `mode.driving`, é possível definir as características físicas do veículo (altura, largura, comprimento, e peso), de forma a evitar trajetos onde estas características poderiam ser um empecilho físico ou legal (e.g., veículo muito pesado).

O trecho na Figura 4 é apenas um pequeno exemplo, e existem diversas outras propriedades que fogem do escopo deste tutorial. Na realidade, os perfis no OSRM podem até mesmo definir

Figura 4: Trecho extraído do perfil padrão `car.lua` do OSRM.

```
properties = {
  weight_name             = 'routability',
  u_turn_penalty          = 20,  -- Tempo em segundos
  traffic_light_penalty = 2      -- Tempo em segundos
},

default_mode              = mode.driving,
turn_penalty              = 7.5,  -- Tempo em segundos

vehicle_height            = 2.5,  -- Altura em metros
vehicle_width             = 1.9,  -- Largura em metros
vehicle_length            = 4.8,  -- Comprimento em metros

vehicle_weight            = 3500, -- Peso em quilogramas
```

Fonte: Elaborado pelos autores.

funções próprias para calcular durações e pesos de trajetos que são usados pelos algoritmos de caminho mais curto. Sugerimos aos interessados a leitura da documentação do OSRM para uma descrição a respeito de todas as funcionalidades. De qualquer forma, o perfil padrão já é capaz de produzir resultados satisfatórios para muitos casos de uso, e a edição de alguns dos seus valores, como os apresentados anteriormente, é bastante simples.

Note, no entanto, que o uso correto de propriedades do perfil que dependem de dados sobre o caminho, requer que o mapa de entrada para o OSRM possua estas informações, o que nem sempre é garantido. No caso de informações faltantes, o usuário pode realizar a edição do mapa do OSM para incluir estes dados, mas isto exige um profundo conhecimento do formato padrão destes arquivos. Para tarefas que não requerem grande precisão, podemos usar diretamente a informação padrão já existente nos mapas disponíveis nos repositórios, sem qualquer edição.

Soluções do PRV também podem ser desenhadas graficamente sobre um mapa real. O OSRM possui uma versão *frontend* disponível em OSRM-frontend (2020), que permite informar uma sequência de coordenadas e, usando o OSRM *backend* para computar os caminhos mais curtos, desenhar a rota no mapa do OSM. Uma versão de demonstração pode ser testada no *website* do projeto disponível em OSRM-website (2020). A Figura 5 apresenta um exemplo de rota de PRV em Barcelona desenhada através dessa ferramenta. Nela podemos perceber um local de origem (verde) e um de destino (vermelho), bem como locais intermediários (cinzas) que são visitados ao longo da rota, como em um PRV tradicional. A partir disso, podemos analisar visualmente as soluções geradas por algum método para uma instância do PRV em mapas reais.

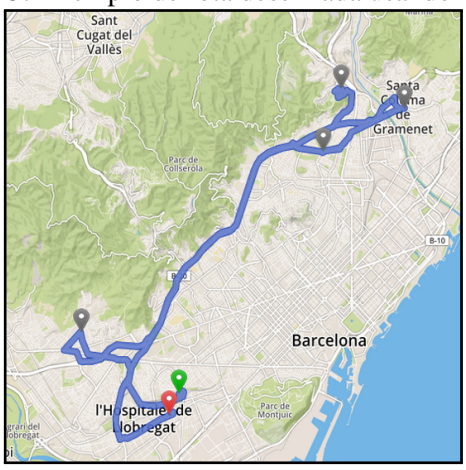
Por fim, ressaltamos que o OSRM é uma ferramenta com desenvolvimento contínuo. Assim, mudanças podem ocorrer no modo de uso da aplicação com a criação de novas versões, o que exige atenção do usuário a respeito da versão OSRM utilizada. Para este tutorial, nos baseamos na versão `5.21` do OSRM-backend (2020).

2.3. Características Específicas

Até agora pudemos obter o grafo básico de qualquer instância do PRV, com os clientes (vértices) e os arcos (caminhos) com seus pesos. Porém, PRVs também possuem características específicas associadas aos clientes, como demanda, janelas de tempo, e pareamento, ou aos veículos, como capacidade, limitação em distância percorrida, ou tempo de viagem máximo.

É claro que, uma vez que o grafo base esteja pronto, todas as outras características específicas

Figura 5: Exemplo de rota desenhada usando OSRM.



Fonte: Elaborado pelos autores.

podem ser criadas como em uma instância de PRV normal. A exceção fica sendo apenas para casos em que os dados utilizados anteriormente já possuem alguma informação a respeito do problema. Como em geral este não é o caso, descrevemos brevemente algumas possibilidades para a geração destas características para finalizar o tutorial.

2.3.1. Demandas de Clientes

A grande maioria dos PRVs da literatura faz uso de demandas nos clientes e capacidades nos veículos. Para a geração destes valores, podemos adotar estratégias como as apresentadas por Uchoa et al. (2017), que incluem todas as demandas unitárias (demandas de valor 1) e demandas selecionadas aleatoriamente dentro de um distribuição uniforme $U[a, b]$ (e.g., demandas entre 1 e 10 poderiam ser geradas com $U[1, 10]$). É claro que os valores de a e b dependem também da capacidade Q do veículo, bem como da variação desejada nas demandas (a e b próximos fornecem uma variação menor). Uma outra alternativa é definir um intervalo dependente do valor de Q.

Por outro lado, os dados de Donovan e Work (2016) para corridas de táxi informam a quantidade de pessoas no veículo para a corrida, o que pode ser interpretado como a demanda. Útil, particularmente, para casos com coleta e entrega, mas que também pode ser adaptado às outras variações do PRV.

2.3.2. Janelas de Tempo

Outra variante envolve janelas de tempo para o atendimento de clientes do tipo $[e_i, l_i]$, indicando o horário mais cedo e_i e mais tarde l_i em que o serviço pode ser iniciado no local i. Uma estratégia possível para gerar estas janelas de tempo foi proposta por Solomon (1987).

Dado um horizonte de planejamento H (i.e., o horário máximo para uma rota terminar, assumindo que as rotas iniciam no tempo 0), e uma *largura* W das janelas de tempo (i.e., a diferença $l_i - e_i = W$), e assumindo que a instância possui um único depósito, podemos gerar os valores e_i e l_i da seguinte forma. Primeiro, selecionamos um valor central C_i, usando o tempo de viagem do depósito até o local i, t_{0i} e vice-versa t_{i0}. Escolhemos C_i de forma aleatória no intervalo $U[t_{0i} + W/2, H - t_{i0} - W/2]$, tal que os extremos desta seleção representam o momento mais cedo e mais tarde que o serviço pode iniciar para garantir uma solução factível do problema. Uma alternativa é selecionar W aleatoriamente e não mantê-lo fixo como proposto por Solomon

(1987). Uma vez definidos C_i e W, podemos calcular a janela de tempo com:

$$e_i = C_i - W/2 \tag{1}$$

$$l_i = C_i + W/2 \tag{2}$$

No caso dos dados de Donovan e Work (2016), podemos utilizar como valor central C_i de cada local (ponto de partida ou destino), o horário exato de início ou fim da corrida, e definir um valor para W. Por exemplo, se selecionarmos um local i de início de uma corrida que começou às $C_i = 8\text{h}00$ e definirmos $W = 0.5\text{h}$, a janela gerada para o local i será $[7\text{h}45, 8\text{h}15]$. Desta forma, conseguimos utilizar a informação original da corrida na geração das instâncias.

2.3.3. Pareamento e Precedência

Problemas de coleta e entrega exigem a geração de pares de locais (p, d) para formar uma requisição do problema, onde a coleta p deve preceder a entrega d em uma mesma rota. Uma forma de realizar o pareamento é de forma completamente aleatória, outra forma é como proposto por Li e Lim (2003), ou seja, resolver uma instância de PRV não pareado primeiro e depois parear clientes que pertencem à mesma rota. Uma terceira forma é parear locais próximos, ou que pertençam a um mesmo agrupamento, de forma a criar uma certa localidade para as requisições (o grau de proximidade pode ser definido por distância ou tempo, por exemplo). Após o pareamento, basta selecionar um dos dois locais para ser a coleta e o outro a entrega.

Se utilizarmos dados como os de Donovan e Work (2016), o pareamento e a precedência já são informados na entrada de dados. Podemos usar exatamente o local de início e de fim da corrida de táxi para esta geração.

3. Considerações Finais

O objetivo deste tutorial foi mostrar como podemos utilizar da grande gama de dados abertos disponíveis a poucos cliques de distância para a geração de instâncias mais realistas para Problemas de Roteamento de Veículos. Claramente, a aplicação não é restrita a PRVs, e pode ser utilizada em qualquer problema onde existe uma rede urbana.

As bases de dados abertos citadas ao longo deste trabalho são apenas sugestões que já foram utilizadas em trabalhos anteriores como Donovan e Work (2017) e Sartori (2019). Na verdade, quaisquer bases que contenham coordenadas (em grande quantidade) para a área de interesse da instância podem ser utilizadas. O mesmo vale para os dados dos mapas, que seguindo o padrão OSM podem inclusive ser modificados pelo usuário para adicionar informações e ainda assim utilizar a ferramenta OSRM já existente para o cálculo dos caminhos. Também vale ressaltar que somada à descrição deste tutorial, uma ferramenta de código aberto para gerar instâncias para o Problema de Coleta e Entrega com Janelas de Tempo está disponível em OVIG (2020). Ela foi desenvolvida para a geração de um novo conjunto de instâncias do problema e tomou como base parte do processo de Uchoa et al. (2017) juntamente com o uso dos dados abertos.

Por fim, esclarecemos ao leitor que, apesar do uso de dados realistas, as instâncias geradas desta forma não são completamente fiéis à qualquer problema de roteamento. Além disso, o processo em si não está livre de falhas, especialmente porque a qualidade das distribuições de clientes e do cálculo de distâncias ou tempos está diretamente relacionada à qualidade dos dados abertos utilizados. Em contrapartida, a dificuldade de gerar instâncias para o PRV também está associada às suas características específicas, para as quais dados abertos raramente estão disponíveis e, portanto, devem ser aproximadas. Em todo caso, esperamos que este tutorial possa servir como base ou inspiração para outros pesquisadores gerarem instâncias para variantes do PRV ainda mais interessantes e diversas, abrindo novas oportunidades de análises a respeito de métodos e soluções dos problemas estudados.

Agradecimentos. Os autores agradecem o apoio financeiro da Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ). Agradecem também o apoio da Universidade Federal do Rio Grande do Sul (UFRGS) ao longo do desenvolvimento da pesquisa que deu origem a este tutorial.

Referências

Dantzig, G. B. e Ramser, J. H. The truck dispatching problem. *Management Science*, v. 6, n. 1, p. 80–91, 1959.

Donovan, B. e Work, D. *New york city trip data (2010 to 2013)*, 2016. Disponível em: `https://doi.org/10.13012/J8PN93H8`. Acesso em: 16/02/2020.

Donovan, B. e Work, D. B. Empirically quantifying city-scale transportation system resilience to extreme events. *Transportation Research Part C: Emerging Technologies*, v. 79, p. 333 – 346, 2017.

Geofabrik. *OpenStreetMap Data Extracts*, 2018. Disponível em: `http://download.geofabrik.de/`. Acesso em: 16/02/2020.

Instituto Brasileiro de Geografia e Estatística. *Censo 2010 – cadastro nacional de endereços para fins estatísticos*, 2010. Disponível em: `https://censo2010.ibge.gov.br/cnefe/`. Acesso em: 16/02/2020.

Laporte, G. Fifty years of vehicle routing. *Transportation Science*, v. 43, n. 4, p. 408–416, 2009.

Li, H. e Lim, A. A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, v. 12, n. 02, p. 173–186, 2003.

Luxen, D. e Vetter, C. Real-time routing with OpenStreetMap data. In: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '11. New York, NY, USA. ACM, 2011. p. 513–516.

OpenAddresses. *The free and open global address collection*, 2017. Disponível em: `https://openaddresses.io/`. Acesso em: 16/02/2020.

OpenStreetMap. *OSM contributors. Planet dump.*, 2017. Disponível em: `https://www.openstreetmap.org`. Acesso em: 16/02/2020.

OSRM-backend. *Repositório Git do OSRM backend*, 2020. Disponível em: `https://github.com/Project-OSRM/osrm-backend`. Acesso em: 16/02/2020.

OSRM-frontend. *Repositório Git do OSRM frontend*, 2020. Disponível em: `https://github.com/Project-OSRM/osrm-frontend`. Acesso em: 16/02/2020.

OSRM-website. *Website do projeto OSRM*, 2020. Disponível em: `https://map.project-osrm.org/`. Acesso em: 16/02/2020.

OVIG. *Repositório Git do projeto Open Source Vehicle Routing Instance Generator*, 2020. Disponível em: `https://github.com/cssartori/ovig`. Acesso em: 16/02/2020.

Sartori, C. S. *The Pickup and Delivery Problem with Time Windows: Algorithms, Instances, and Solutions*. 101 f. Dissertação (Mestrado em Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre-RS, 2019.

Schneider, A. *GPS* Visualizer, 2003. Disponível em: `https://www.gpsvisualizer.com/`. Acesso em: 16/02/2020.

Solomon, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, v. 35, n. 2, p. 254–265, 1987.

Toth, P. e Vigo, D. (eds.) *Vehicle Routing*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T. e Subramanian, A. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, v. 257, n. 3, p. 845–858, 2017.