MATHEUS BENTO FERNANDES

DEFINIÇÃO DE PONTOS DE PARADA PARA VEÍCULOS DE TRANSPORTE COLETIVO ESCOLAR

LAVRAS - MG
2021

MATHEUS BENTO FERNANDES

DEFINIÇÃO DE PONTOS DE PARADA PARA VEÍCULOS DE TRANSPORTE COLETIVO ESCOLAR

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

Prof. Dr. André Vital Saúde
Orientador

LAVRAS - MG
2021

Dedico este artigo ao meu pai José Raimundo Fernandes e minha mãe Aparecida Regina Bento por todo suporte durante meu período de graduação, o que possibilitou este momento de acontecer.

AGRADECIMENTOS

À Universidade Federal de Lavras - UFLA e ao Departamento de Ciência da Computação - DCC por todo conhecimento adquirido durante o período de graduação.

Ao Professor André Vital Saúde por toda disponibilidade, paciência e ajuda durante o processo de desenvolvimento deste trabalho.

À Fapemig e CNPq pelo financiamento de projetos de pesquisa que permitiram a execução deste artigo.

À Secretaria da Educação do Estado de São Paulo pelo fornecimento dos dados experimentais.

Sumário

Definição dos pontos de parada para veículos de transporte coletivo escolar

Matheus B. Fernandes[1], **André V. Saúde**[2]

[1]Departamento de ciência da computação – Universidade Federal de Lavras (UFLA)
Caixa Postal 3037 – CEP 37200-900 – Lavras – MG – Brasil

[1]matheus.fernandes1@estudante.ufla.br, [2]saude@ufla.br

Abstract. *This article addresses the issue of defining bus stops for passenger boarding and disembarking, aiming to offer a solution to the transport of students, one of the various types of on-demand public transport. The maximum walking distance constraint from the student's home to the bus stop is considered. The strategy used considers the discretization of the real data collected, aiming at simplifying the problem, in addition is presented an algorithm with a greedy approach to define bus stop in order to minimize the number of stops. The solution was tested with real student data, and it was possible to see good results in polynomial time.*

Resumo. *Este artigo aborda a questão da definição dos pontos de paradas de ônibus para o embarque e desembarque de passageiros, visando oferecer uma solução ao transporte de alunos, um dos vários tipos de transporte coletivo sob demanda. É considerada a restrição de máxima distância a pé da casa do aluno para ponto de ônibus. A estratégia utilizada leva em consideração a discretização dos dados reais coletados, visando uma simplificação do problema, além de apresentar um algoritmo de abordagem gulosa para definir pontos de parada de ônibus com o objetivo de minimizar a quantidade de pontos. A solução foi testada com dados reais de estudantes, e foi possível constatar bons resultados em tempo polinomial.*

1. Introdução

Por transporte coletivo sob demanda entende-se o transporte de pessoas de forma planejada, quando todas as origens, destinos e horários das viagens requeridas pelos usuários são conhecidas antes da definição dos veículos e rotas que irão prestar o serviço. Vários exemplos desse tipo de transporte podem ser descritos, por exemplo, o problema de roteamento de ônibus escolar (SBRP - School Bus Routing Problem). Em um grande número de estudos o SBRP é descrito como uma importante aplicação em mundo real do VRP (Vehicle Routing Problem). Enquanto um problema típico de roteamento de veículos lida principalmente com o transporte de carga, o SBRP está relacionado com o transporte de pessoas, neste caso alunos. A maioria das publicações de roteamento de ônibus escolares concentram-se na formulação de restrições e objetivos extras levando em consideração alguns fatores relacionados ao aluno. Por essas razões, os problemas de SBRP costumam ser mais complicados do que um VRP [Worwa 2014].

De acordo com [Desrosiers et al. 1981] o SBRP pode ser formulado em cinco subproblemas: *data preparation*, *bus stop selection*, *bus route generation*, *school bell time*

adjustment e *route scheduling*. Na maioria das abordagens existentes, esses subproblemas são considerados separadamente e sequencialmente. Embora estes subproblemas não sejam independentes, mas estejam altamente inter-relacionados, eles são tratados de forma separada devido à complexidade e ao tamanho do problema. Além disso, na maioria da literatura, apenas algumas partes do SBRP são consideradas [Park and Kim 2010]. Dessa forma este artigo considera apenas a etapa de seleção de parada de ônibus (*bus stop selection*), por já ser um problema complexo de ampla abordagem.

A solução apresentada neste trabalho tem como objetivo a criação de pontos de ônibus a partir de dados reais sobre os locais de residência dos estudantes, levando em conta que os resultados obtidos podem ser utilizados mais tarde nas outras etapas do SBRP, em combinações com outros algoritmos. Desta forma é considerado que os alunos possuem uma restrição máxima de distância que podem caminhar até algum ponto.

A implementação é dividida em quatro algoritmos, que visam simplificar o problema da geometria contínua para a geometria discreta, fazendo com que exista uma redução na complexidade geral do problema e a etapa de seleção de pontos seja mais escalável para exemplos do mundo real. A solução utilizada, caracteriza-se, ainda, por utilizar uma abordagem gulosa que geram os locais de parada.

Com as análises realizadas é possível demonstrar que o algoritmo possui complexidade polinomial, e os resultados obtidos através dos experimentos mostram uma solução com pouca variabilidade devido a aleatoriedade do algoritmo, assim as execuções que apresentam os melhores resultados e as que apresentam os piores possuem respostas bastante próximas. Entretanto não foi possível encontrar qualquer relação com os resultados do algoritmo e o ótimo global para o problema.

A literatura sobre o SBRP é vasta incluindo as mais diversas abordagens para o subproblema de seleção das paradas de ônibus. Alguns trabalhos abordam vários subproblemas juntos da seleção de paradas, como [Faraj et al. 2014] que considera as etapas de preparação dos dados, passando pela seleção dos pontos e chegando no algoritmo de roteamento. Já outras publicações não consideram o roteamento de veículos e limitam-se a explorar apenas a seleção dos pontos, como [Sarubbi et al. 2016] que utiliza uma abordagem GRASP para resolver o problema de seleção dos pontos. Alguns artigos nem mesmo consideram o BSS (bus stop selection), e assumem que os locais dos pontos já serão informados por órgãos responsáveis desde o inicio [Galdi and Thebpanya 2016].

Este artigo é organizado da seguinte forma: a Seção 2 faz uma análise do referencial teórico; a Seção 3 descreve a metodologia, os diferentes passos utilizados na solução; a Seção 4 apresenta os resultados encontrados; a Seção 5 conclui o trabalho; e a Seção 6 faz os agradecimentos às instituições importantes para a execução deste trabalho.

2. Referencial Teórico

Como descrito anteriormente, a literatura sobre a definição dos pontos de paradas de ônibus é abrangente, normalmente acompanhada de um problema de roteamento de veículos ou suas variações como o SBRP. Desta forma este trabalho utiliza como base [Ellegood et al. 2020] para ter uma visão ampla da área de pesquisa em questão. Em seu artigo, Ellegood et al. baseiam-se fortemente em [Park and Kim 2010] para realizar um trabalho de revisão da literatura a cerca do SBRP, listando sessenta e quatro publicações, classificando-as, assim, em sete categorias que visam mostrar as características

importantes das pesquisas realizadas na área. A primeira dessas categorias considera os subproblemas do SBRP.

Dos subproblemas a seleção dos pontos de parada (*bus stop selection* - BSS) identifica locais de parada de ônibus para cada estudante, geralmente com base em uma lista aprovada de locais candidatos a ponto de ônibus. Esses locais são fornecidos pelo distrito, podem ser baseados em características do lado da rua (por exemplo, aspectos de visibilidade das ruas que tornam os pontos de ônibus desejáveis e seguros), características do ponto de ônibus (por exemplo, recursos fora da rua, como espaço disponível para esperar o ônibus) e nas características das rotas dos estudantes (por exemplo, ao longo do caminho de um estudante de sua residência até um ponto de ônibus) [Ellegood et al. 2020]. Dezessete publicações, quase um quarto das pesquisas recentes sobre SBRP consideram a seleção de pontos de ônibus, onde quase sempre são examinadas em uma combinação com o roteamento dos ônibus escolares, sendo [Sarubbi et al. 2016] e [Galdi and Thebpanya 2016] exceções que consideram a seleção dos pontos como problemas independentes. Geralmente, o subproblema de seleção de parada de ônibus é resolvido primeiro, pois a localização e a demanda de cada ponto são entradas para o problema de roteamento [Ellegood et al. 2020].

Realizando uma análise dos artigos levantados por [Ellegood et al. 2020] é possível perceber que dos dezessete que citam o subproblema de BSS, apenas onze consideram a restrição MWT (tempo máximo de caminhada ou distância) onde é estabelecida uma distância máxima que os estudantes podem andar até o ponto de ônibus. Além disso, três publicações podem ser consideradas trabalhos que negligenciam a otimização do número de paradas, desta forma estes não serão analisados. Assim apenas oito artigos são utilizados na análise deste trabalho. Essas publicações podem lidar com ambientes rurais, urbanos ou ambos e ainda possuir objetivos diferentes a serem minimizados: SWD (distância total do aluno a pé) considera minimizar o tempo de caminhada de um estudante até um ponto de ônibus, NS (número de paradas) considera minimizar a quantidade de pontos e SBS (ponto de ônibus compartilhado) em que apenas [Bögl et al. 2015] considera minimizar os pontos compartilhados para um distrito escolar com uma política de carga mista de transbordo, acreditando que isso reduziria o número de transferências de ônibus, resultando em uma rede de transporte mais eficiente [Ellegood et al. 2020]. Entretanto [Ellegood et al. 2015] contesta a hipótese utilizada por Bögl et al, mostrando que, para um problema de carga mista, uma abordagem que pode melhorar a eficiência de uma rede de ônibus escolar é maximizar pontos de ônibus compartilhados. Desta forma este artigo não considera o objetivo SBS.

Considerando as informações apresentadas é possível descrever este trabalho considerando que a restrição MWT deve ser atribuída individualmente, permitindo lidar com uma certa diversidade de passageiros, inclusive aqueles com restrições de mobilidade e que devam ser atendidos na porta de suas casas. Isso faz com que o objetivo SWD não seja considerado, porém parte-se da hipótese de que as paradas são o que mais aumentam o tempo de viagem e por conta disso é buscado ter o mínimo pontos de ônibus, assim é assumido o objetivo NS. Como o problema abordado neste artigo não vincula paradas a veículos ou rotas, portanto as paradas são todas compartilháveis (SBS) e podem, inclusive, ser servidas por um ou mais veículos no momento do roteamento.

A partir da definição dos objetivos desta solução é possível fazer uma comparação

Tabela 1. Publicações com restrição MWT e objetivos.

Referência	Ambiente do Serviço	Objetivos
[Martínez and Viegas 2011]	Urbano	SWD
[Riera-Ledesma and Salazar-González 2012]	Urbano	SWD
[Riera-Ledesma and Salazar-González 2013]	Urbano	SWD
[Pérez-Rodríguez and Hernández-Aguirre 2016]	Ambos	NS
[Sarubbi et al. 2016]	Rural	NS
[Galdi and Thebpanya 2016]	Ambos	NS
[Bögl et al. 2015]	Desconhecido	SWD, NS, SBS
[de Souza and Siqueira 2010]	Urbano	SWD

com os artigos da Tabela 1 que foi construída baseada em [Ellegood et al. 2020] e inclui apenas artigos com restrição MWT e objetivos SWD, NS ou SBS. Nota-se que apenas quatro publicações tem como objetivo minimizar a quantidade de pontos de ônibus (NS). [Pérez-Rodríguez and Hernández-Aguirre 2016] considera uma lista de potenciais pontos de ônibus (por pontos em potenciais entende-se todos aqueles que respeitam a restrição MWT para cada estudante), e através de um processo interativo adiciona estudante a essas paradas, sempre respeitando a capacidade máxima do ponto.

[Sarubbi et al. 2016] primeiramente cria pontos equidistantes ao longo da rede rodoviária (sobre as ruas), a distância de um desses pontos a outro é definida por um parâmetro alfa e é medido em kilômetro. Esses pontos serão os candidatos a serem paradas de ônibus. Em seguida, é criada uma topologia baseada nas informações geométricas dos dados georreferenciados. Os estudantes são então projetados aos pontos mais próximos de suas posições georreferenciadas, assim é calculada a distância de cada ponto que pode ser uma parada de ônibus para todos os estudantes. Por fim é utilizada uma heurística construtiva que a cada iteração escolhe um novo ponto de ônibus até que todos os estudantes estejam associados a uma dessas paradas.

[Galdi and Thebpanya 2016] utilizam dados fornecidos pela Howard County Pupil Transportation Office para implementar uma heurística baseada em GIS que combina uma abordagem de GIS matemático com um processo manual, visando identificar paradas de ônibus desnecessárias, otimizando o tempo de viagem e custos de transporte. Deste modo várias restrições são consideradas: a distância mínima entre pontos de ônibus, presença de faixa de pedestres, rejeitar ruas não seguras e a distância de estudantes até o ponto de ônibus, com um limite de 0,4 milhas que um estudante pode caminhar até alguma parada.

[Bögl et al. 2015] abordam a questão do SBRP em que as crianças podem fazer a transferência entre veículos. Consideram três alternativas para resolver o problema de atribuição de aluno e seleção da parada de ônibus: minimizando a distância para os destinos dos alunos, isto é, os alunos são atribuídos aos pontos de parada que estão localizados na direção de seus destinos. Minimizar a fragmentação dos pontos de ônibus, visa alocar alunos de diferentes escolas em diferentes pontos de ônibus assim, obtendo redes de transporte para as diferentes escolas da forma mais independente possível. Como terceira alternativa Bögl et al. levam em conta a estratégia de minimizar o número de pontos de

ônibus.

Consideram ainda que os subproblemas de seleção dos pontos de parada, roteamento de ônibus, agendamento de ônibus e ajustamento com o tempo de inicio da escola estão interconectados e são resolvidos de uma maneira integrada, entretanto considera uma lista inicial de pontos de ônibus no qual os estudantes serão alocados respeitando a máxima distância que eles podem caminhar.

Embasado no artigo de revisão não foi encontrado um trabalho que tenha abordado o problema BSS com objetivo NS e restrição MWT que utilizasse uma abordagem gulosa baseada em geometria discreta. Dessa forma é possível concluir que a proposta deste artigo é original.

2.1. Tecnologias

Os dados de mapas utilizados neste trabalho foram coletados a partir do projeto $OpenStreetMap$ (OSM), construído por uma comunidade que contribue e mantém dados sobre estradas, trilhas, cafés, estações ferroviárias e muito mais, em todo o mundo [OpenStreetMap 2004]. Além disso, para a manipulação facilitada dos mapas do OSM em Java foi utilizado o $GraphHopper$ (GH) open source [GraphHopper 2014].

O $GraphHopper$ é responsável por criar representações dos mapas do OSM para serem utilizadas por algoritmos em grafos. Assim, algumas funções importantes do GH são utilizadas para a implementação desta solução, como o algoritmo A* de busca entre dois pontos no mapa [Zeng and Church 2009]. O GH também implementa o ajuste de ponto: dada uma localização qualquer contendo sua latitude e longitude, encontra-se o par (latitude e longitude) mais próximo que existe sobre o mapa do OSM.

3. Metodologia

Esta seção descreve em detalhes os algoritmos desenvolvidos para resolver o problema de seleção de pontos de ônibus. Descreve também, os passos necessários desde as aproximações feitas até a solução final.

3.1. Aproximação

A proposta para a resolução do problema de BSS assume duas aproximações importantes, que são consideradas a seguir.

Primeiramente, a solução passa por uma aproximação de um problema de geometria contínua para um problema de geometria discreta. A razão para isso é que a solução utiliza ordenação, e o custo computacional da ordenação de números inteiros pode ser reduzido a $\mathcal{O}(N + K)$ pelo counting sort (mais detalhes sobre a complexidade do counting sort estão na subseção 3.2.2), quando comparado aos típicos algoritmos de ordenação de números reais. Logo é necessário entender o que significa essa aproximação.

A Terra é um esferoide oblato, ou seja, uma esfera ligeiramente achatada nos polos, de tal forma que o esferoide seja apenas 0,3% diferente de uma esfera perfeita [David D. Pollard 2005, p. 28]. Desta forma este trabalho faz a primeira aproximação de considerar a Terra uma esfera, com isso é possível considerar que a distância entre dois pontos será obtida por cálculos simples de geometria plana de circunferências.

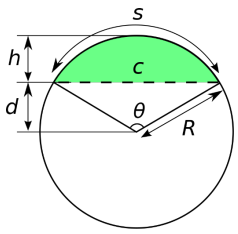


Figura 1. Segmento circular

O raio da Terra pode ser considerado como sendo de 6378,1km [Mamajek et al. 2015, p. 28]. Assim pela Figura 1, é possível mostrar que $\frac{c/2}{R} = \sin(\frac{\theta}{2})$, o que dá $\theta = 2 * \arcsin(\frac{c/2}{R})$. É possível ter ainda que $s = R * \theta = 2 * R * \arcsin(\frac{c/2}{R})$. Logo, se considerar que para $c = 100km$, então $s = 100,001024308668km$, ou seja, o arco é pouco mais que 0,001% maior do que a corda. Essa distância de 100km cobre toda a região metropolitana de São Paulo, como mostra a Figura 2.

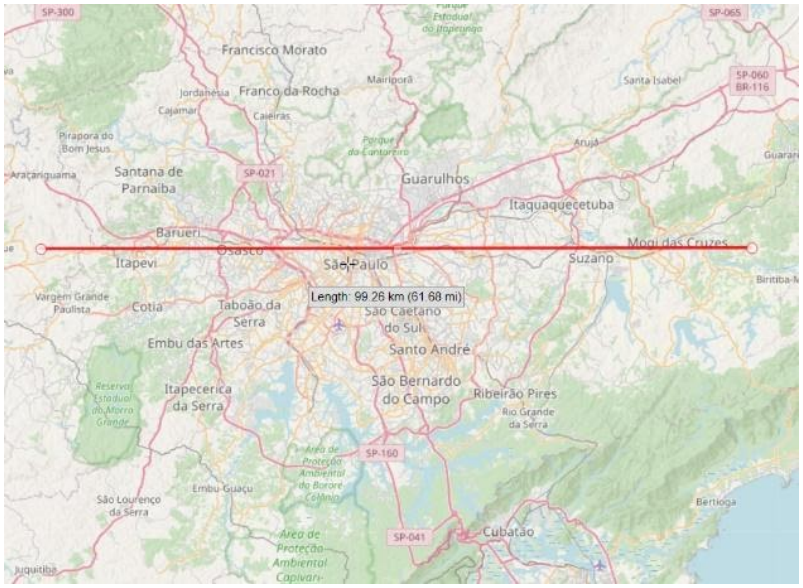


Figura 2. Região metropolitana de São Paulo

Se for considerado $c = 600km$, então $s = 600.221465575776km$, ou seja, o arco é pouco mais de 0,04% maior do que a corda. Essa distância de 600km cobre o espaço de Osaka a Tokio, uma das maiores megalópoles do mundo, como mostra a Figura 3.

Desta forma é possível assumir a segunda aproximação: em uma área quadrada que cobre a região metropolitana de uma cidade muito grande, a Terra pode ser considerada plana, com um erro menor que 0,05%.

3.2. Descrição da solução

A solução desenvolvida para o problema é um processo dividido em quatro etapas:

- Discretização da entrada;
- Ordenação da entrada discreta;
- Construçao do grafo de vizinhanças (Maxonfoot Graph);

Figura 3. Distância de Osaka a Tokio

- Agrupamento.

Entretanto, antes de descrever cada uma dessas etapas são necessárias algumas definições:

1. As residências e o ponto de destino dos passageiros são georreferenciados e localizados por um par de coordenadas de latitude e longitude $\langle lat, lon \rangle$, que podem ser exibidos em ferramentas de mapas, como mostra a Figura 4.
2. $maxonfoot$: é a máxima distância, em metros, que pode ser percorrida à pé por um passageiro da sua residência até o ponto de parada de embarque, ou do ponto de parada de desembarque até o destino (maximum on foot distance);
3. $Point$: é uma tupla $\langle lat, lon, mof \rangle \in \langle \mathbb{R}, \mathbb{R}, \mathbb{Z} \rangle$, que representa a localização da residência e a $maxonfoot$ (mof) de um passageiro;
4. $[Point]$: é uma lista de Point;
5. $pixelsize \in \mathbb{R}_+^*$: Valor medido em graus, é um ângulo na longitude e também na latitude, define uma área na superfície terrestre que pode ser aproximada de um quadrado que será chamado de pixel com tamanho $pixelsize \times pixelsize$. Estes pixels farão parte de uma matriz responsável por simplificar o problema de geometria contínua para um problema de geometria discreta.
6. $dPoint$: é uma tupla $\langle dlat, dlon, p \rangle$, onde $dlat, dlon \in \mathbb{Z}$ e p é um Point.
7. $G(V, E)$: é um grafo, em que V representa o conjunto de vértices e E as arestas.
8. $Cluster$: é o par $(s, [dPoint])$, onde s é o local do ponto de ônibus e $[dPoint]$ indica a lista de locais associados a parada (cluster).

É importante ressaltar que o problema abordado não diferencia os pontos de parada como sendo de embarque ou desembarque, visto que não está sendo feita a otimização de rotas ou capacidade de veículos. Além disso, os pontos são considerados de

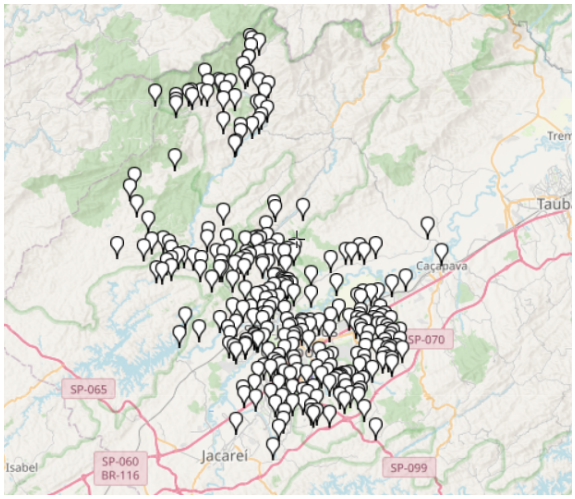


Figura 4. Pontos Latitude e Longitude no mapa

capacidade infinita, podendo servir como local de embarque ou desembarque. O que importa para o problema de localização de pontos de parada é que cada residência ou ponto de destino deva possuir um ponto de parada a uma distância máxima à pé, no mapa, de seu valor $maxonfoot$.

3.2.1. Etapa 1: Discretização da entrada

A primeira etapa é a discretização da entrada, que é feita com o uso do Algoritmo 1.

Algoritmo 1: discretize

Entrada: $points : [Point]; pixelsize \in \mathbb{R}_+^*$
Saída: $dpoints : [dPoint]; nLatPixels, nLonPixels \in \mathbb{N}$

início
  $MinLat \leftarrow \min\{lat, \langle lat, lon, mof\rangle \in points\}$
  $MinLon \leftarrow \min\{lon, \langle lat, lon, mof\rangle \in points\}$
  $MaxLat \leftarrow \max\{lat, \langle lat, lon, mof\rangle \in points\}$
  $MaxLon \leftarrow \max\{lon, \langle lat, lon, mof\rangle \in points\}$
  $nLatPixels \leftarrow \lfloor \frac{MaxLat - MinLat}{pixelsize} \rfloor$
  $nLonPixels \leftarrow \lfloor \frac{MaxLon - MinLon}{pixelsize} \rfloor$
  $dpoints \leftarrow []$
  para *cada* $p \in points$ **faça**
    $\langle lat, lon, mof\rangle \leftarrow p$
    $dlat \leftarrow \lfloor \frac{lat - MinLat}{pixelsize} \rfloor$
    $dlon \leftarrow \lfloor \frac{lon - MinLon}{pixelsize} \rfloor$
    $dpoints.append(\langle dlat, dlon, p\rangle)$
  fim
fim

Nesta primeira etapa, o algoritmo detecta o retângulo delimitador mínimo, ou seja,

o menor retângulo englobando todos os pontos de $points$, como mostra a Figura 5. Tal retângulo é definido pelas latitudes e longitudes minímas e máximas da lista de pontos, representadas por $MinLat$, $MinLon$, $MaxLat$ e $MaxLon$. Este processo é executado em complexidade $\mathcal{O}(N)$, onde N é o número de pontos na lista.

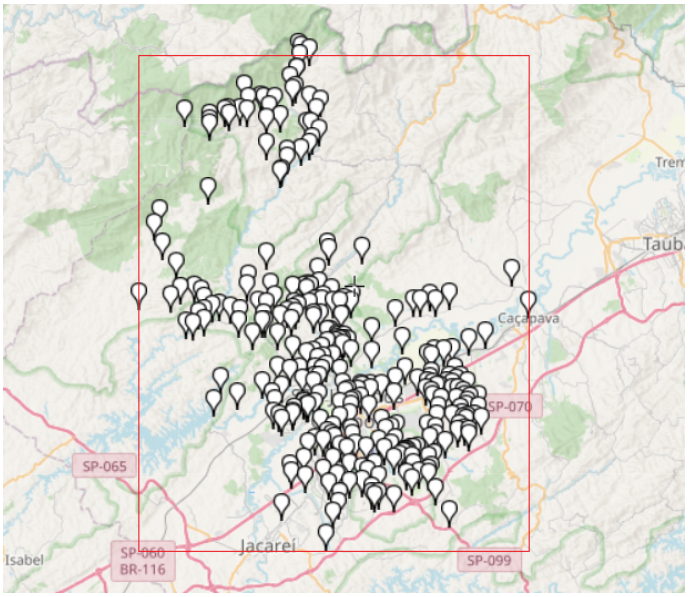


Figura 5. menor retângulo formado por coordenadas de $points$

Em seguida, o algoritmo calcula $nLatPixels$ e $nLonPixels$, essas variáveis indicam os valores máximos que os pares de coordenadas $\langle dlat, dlon\rangle$ (projeções de coordenadas $\langle lat, lon\rangle$ para o espaço discreto) poderão assumir (operação executada em tempo $\mathcal{O}(1)$). Em seguida o processo feito no cálculo anterior é generalizado para converter todos os pontos p de $points$ em coordenadas discretas compreendidas no intervalor de $[0, nLonPixels]$, para a longitude, e $[0, nLatPixels]$, para a latitude (tempo $\mathcal{O}(N)$).

A lista resultante $dpoints$ é, então, uma lista de pontos de origem ou destino, com coordenadas discretas. A partir dessa lista pode ser construída uma matriz de dimensões $nLonPixels \times nLatPixels$ em que o valor dos elementos $\langle i, j\rangle$ podem ser pontos $\langle dlat, dlon, p\rangle \in dpoints$ tais que $\langle dlat, dlon\rangle = \langle i, j\rangle$ e a quantidade de elementos efetivamente preenchidos dessa matriz é igual a quantidade de pontos em $dpoints$. Tal matriz é uma imagem digital e pode ser processada por algoritmos já conhecidos de processamento digital de imagens e geometria discreta.

3.2.2. Etapa 2: Ordenação da entrada discreta

A segunda etapa é a ordenação da entrada discreta, conforme descrito no Algoritmo 2. Como todas as coordenadas são agora discretas, com valores no intervalo $[0, nLonPixels]$, na longitude, e valores no intervalo $[0, nLatPixels]$, na latitude, é possível ordenar a lista $dpoints$, com base nas coordenadas dos pontos, utilizando o algoritmo Counting Sort [Cormen et al. 2009, p. 194], que tem complexidade $\mathcal{O}(N + K)$ em que N é o tamanho da lista $dpoints$ e K é o delta entre a maior e menor chave.

Embora K indique um comportamento exponencial, o retângulo máximo que a solução considera, levando em conta o exemplo que cobre toda a região metropolitana de São Paulo (Figura 2), possui uma área de 100km x 100km, ou seja, K é de, no máximo 10 mil. Assim para entradas pequenas, é possível considerar a utilização de outro algoritmo, mas este artigo visa o problema para entradas grandes, desta forma o counting sort se torna uma boa alternativa.

Algoritmo 2: discreteSort

Entrada: $dpoints : [dPoint]; nLatPixels, nLonPixels \in \mathbb{N}$
Saída: $\langle vetorWE, vetorEW, vetorNS, vetorSN \rangle$

1 **início**
2 $vetorWE \leftarrow countingSort(dpoints, nLonPixels)$, $dlon$ crescente (oeste para leste)
3 $vetorEW \leftarrow countingSort(dpoints, nLonPixels)$, $dlon$ decrescente (leste para oeste)
4 $vetorNS \leftarrow countingSort(dpoints, nLatPixels)$, $dlat$ crescente (norte para sul)
5 $vetorSN \leftarrow countingSort(dpoints, nLatPixels)$, $dlat$ decrescente (sul para norte)
6 **fim**

3.2.3. Etapa 3: Construção do grafo de vizinhanças (MaxonfootGraph)

O grafo de vizinhanças utilizado nesta solução é chamado de $MaxonfootGraph$. Trata-se de um grafo especificamente definido para este trabalho.

O $MaxonfootGraph$ utiliza todos os pontos de origem ou destino como sendo os vértices, e as arestas são definidas a partir dos conjuntos de vértices que têm o potencial de compartilhar o mesmo ponto de parada. O $MaxonfootGraph$ é, portanto, um grafo não dirigido que simplesmente conecta cada ponto de origem ou destino aos pontos que possuem potencial de compartilhar uma parada.

O objetivo de se construir esse grafo é reduzir o custo de busca por combinações de pontos para se produzir um agrupamento, que será o ponto de parada. Dois pontos são considerados potencialmente agrupáveis (possuem potencial de compartilhar uma parada) se a distância Euclidiana entre eles for menor do que a soma de seus respectivos valores de $maxonfoot$. Se esta condição não for satisfeita, será impossível encontrar no mapa um ponto de parada que possa ser compartilhado. Isso pode ser melhor explicado de forma ilustrada.

Considerando a Figura 6a, os pontos verdes tiveram a máxima distância a pé ($maxonfoot$) definida como $300m$. Os pontos vermelhos tiveram $maxonfoot$ definida como $200m$. A Figura 6b mostra a distância euclidiana entre dois pontos do exemplo, que é de $580m$. Só é possível que exista uma parada de ônibus podendo ser compartilhada entre dois pontos se a distância euclidiana for menor do que a soma de seus respectivos $maxonfoot$. No exemplo da Figura 6b, $580m < 300m + 300m$, atendendo ao critério de potencialmente agrupáveis. Olhando o mapa observamos que existe um rio entre esses dois pontos e que as distâncias à pé no mapa são muito diferentes da distância Euclidi-

ana. Porém, analisando apenas cálculos geométricos não é possível fazer essa observação. Neste caso, esses dois pontos serão considerados potencialmente agrupáveis e haverá uma aresta ligando-os no $MaxonfootGraph$.

Já na Figura 6c a distância entre os dois pontos é menor ($440m$), porém os valores de $maxonfoot$ também são menores, e como $440m > 200m + 200m$, não existe a possibilidade desses dois pontos compartilharem uma mesma parada. Por fim, na Figura 6d é possível notar um exemplo com diferentes valores de $maxonfoot$. A distância total é de $540m$, mas $540m > 300m + 200m$, portanto não existe a possibilidade desses dois pontos compartilharem uma mesma parada.

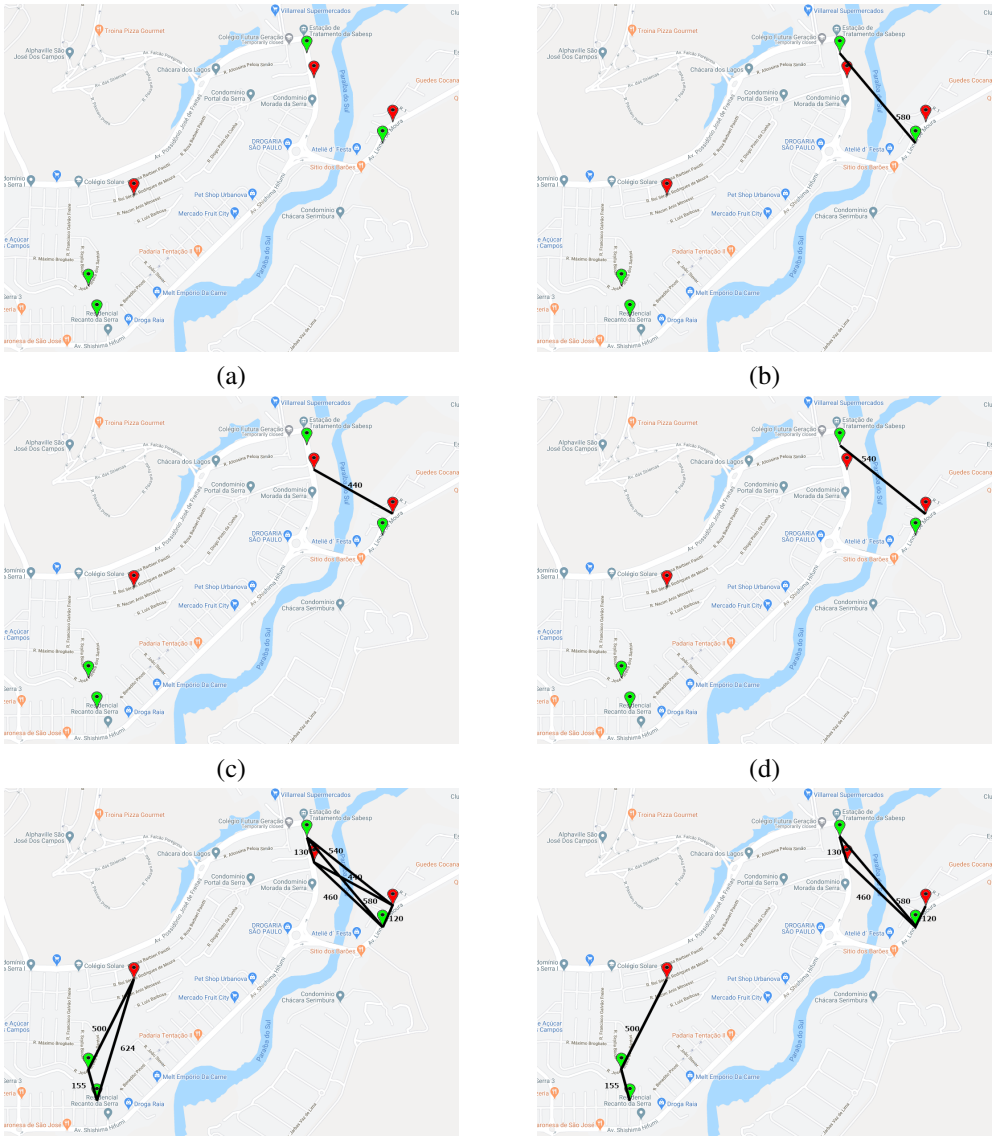


Figura 6. Construção do MaxonfootGraph.

A Figura 6e mostra todas as distâncias (exceto as que são claramente muito grandes) entre os pares de pontos que aparecem no mapa. A partir dessas distâncias e com base nos valores de $maxonfoot$, vamos manter conectados apenas os pares de pontos

que podem vir a compartilhar uma parada. O resultado é exibido na Figura 6f. Assim, a criação do $MaxonfootGraph$ pode ser definido pelo Algoritmo 3.

Inicialmente é criada a lista V contendo todos os pontos discretos de $dpoints$, essa lista constituirá de todos os vértices do $MaxonfootGraph$. Em seguida, é verificado, para cada $v \in V$ quais pontos $p \in dpoints$ respeitam a condição em que a distância euclidiana $d(v, p)$ será menor que a soma de seus respectivos $maxonfoot$, para os pares de pontos que respeitarem a condição, eles serão adicionados ao conjunto de arestas do grafo, além disso será adicionado também o peso dessa aresta, valor dado pela distância euclidiana. Assim, pode-se concluir pelo algoritmo que sua complexidade é da ordem de $\mathcal{O}(N^2)$.

Algoritmo 3: getMaxonfootGraph

```
Entrada: dpoints : [dPoint]
Saída: maxOnfootGraph
1  início
2      V ← dpoints
3      E ← []
4      para cada v ∈ V faça
5          para cada p ∈ dpoints faça
6              se d(v, p) < v.maxonfoot + p.maxonfoot então
7                  E.append((v, p, d(v, p)))
8              fim
9          fim
10     fim
11     maxOnfootGraph ← G(V, E)
12 fim
```

3.2.4. Etapa 4: Agrupamento

Finalmente, nesta última etapa do processo é feito o agrupamento com o algoritmo de clusterização local. Até o momento todas as fases serviram como pré-processamento. A solução para o problema de negócio, de fato, é dada nesta subseção.

A criação dos clusters locais é definida pelo Algoritmo 4 e tem como saída os locais de paradas de ônibus.

Inicialmente o algoritmo $clusterize$ faz uma chamada ao Algoritmo 1 ($discretize$) convertendo as coordenadas reais em discretas, esse passo é indicado pela conversão da lista $points$ em $dpoints$. Em seguida a função $discreteSort$ (Algoritmo 2) é responsável por trazer a lista de pontos ordenados de oeste para leste (vWE), leste para oeste (sEW), norte para sul (vNS) e sul para norte (vSN). Além disso é também criado o $MaxonfootGraph$, a partir da lista de coordenadas discretas utilizando o Algoritmo 3.

A sequência do algoritmo tem por pressuposto que todos os indivíduos devem ser transportados, mesmo o mais distante do destino. Aqui entra a visão gulosa da solução.

Considerando os indivíduos dos extremos leste, oeste, norte ou sul como os que

não podem ser "atraídos" para pontos de parada, mas sim, como os que "atrairão" os pontos de parada. O pensamento é que deve-se pegar o indivíduo mais distante e, por isso, é buscado, para cada indivíduo dos extremos, os estudantes que possam compartilhar com ele o ponto de parada. Dessa forma é feita a aproximação de que indivíduos na máxima ou mínima latitude ou longitude discreta são os extremos, e por eles é iniciada a criação de pontos de ônibus.

Algoritmo 4: clusterize

Entrada: $points : [Point]; pixelsize \in \mathbb{R}_+^*$
Saída: $stops : [Cluster]$

início
  $dpoints, nLat, nLon \leftarrow discretize(points, pixelsize)$
  $vWE, vEW, vNS, vSN \leftarrow discreteSort(dpoints, nLat, nLon)$
  $mofGraph \leftarrow getMaxonfootGraph(dpoints)$
  $visited \leftarrow []$
  $stops \leftarrow []$
  repita
    $vec \leftarrow random(vWE, vEW, vNS, vSN)$
    $p \leftarrow vec.peek()$
    $vWE.remove(p)$
    $vEW.remove(p)$
    $vNS.remove(p)$
    $vSN.remove(p)$
    se $p \notin visited$ **então**
      $s \leftarrow barycenterAdjust(p)$
      $visited.append(p)$
      $neighbors \leftarrow maxOnfootGraph.sortedNeighbors(p)$
      $Lc \leftarrow []$
      $Lc.append(p)$
      repita
        $closest \leftarrow neighbors.peek()$
        $newS \leftarrow barycenter(Lc \cup closest)$
        $newS \leftarrow barycenterAdjust(newS)$
        se $\forall c_i \in Lc \cup closest,\ dMap(c_i, newS) < c_i.maxonfoot$ **então**
          $Lc.append(closest)$;
          $visited.append(closest)$
          $mofGraph.remove(closest)$
          $s \leftarrow newS$
        fim
      até $neighbors \in \emptyset$;
      $mofGraph.remove(p)$
      $stops.append((s, Lc))$
    fim
  até $vWE = \emptyset \wedge vEW = \emptyset \wedge vNS = \emptyset \wedge vSN = \emptyset$;
fim

Tendo em vista a ideia de começar por uma das extremidades de modo aleatório é

selecionado um vetor dentre vWE, sEW, vNS e vSN. O método $vec.peek()$ seleciona o primeiro elemento (p) do vetor vec (este indica o ponto mais ao extremo, seja pela longitude ou latitude) e o remove das listas de pontos (linhas 8 a 13).

Se p não foi visitado ainda, ele é marcado como visitado e os vizinhos de p no $MaxonfootGraph$ são ordenados pelo peso das arestas que representam a distância euclidiana até p, em ordem crescente, formando a lista de vizinhos $neighbors$ (linha 17).

Em seguida, é criado o cluster Lc, contendo apenas p, e enquanto a lista de vizinhos não for vazia, extrai-se $closest$, o vizinho mais próximo de p, indicado por $neighbors.peek()$. Calcula-se o baricentro euclidiano discreto dos pontos da lista $Lc \cup closest$ (linha 22). Entretanto, o baricentro euclidiano pode estar posicionado no mapa fora de uma via acessível de carro. A função $barycenterAdjust(newS)$ ajusta o baricentro ao mapa de vias de carro, retornando a coordenada do ponto, sobre uma rua do mapa, e que seja o mais próximo a $newS$ (linha 23).

Com o novo baricentro ajustado, a distância de cada ponto de $Lc \cup closest$ até o baricentro deve ser verificada, para garantir a restrição de máxima distância à pé ($c_i.maxonfoot$, linha 24). Para esta verificação é utilizada a função $dMap$, que calcula a distância real à pé, isto é, a distância sobre os caminhos do mapa que podem ser percorridos à pé.

Caso algum ponto da lista tenha como $maxonfoot$ um valor menor que a distância real calculada por $dMap$, esse ponto não poderá fazer parte do cluster Lc, e continua considerado não visitado. Por outro lado, caso não exista um valor de distância maior que o $maxonfoot$, então adiciona-se o ponto ao cluster e à lista de visitados, além de removê-lo do $MaxonfootGraph$. Em outras palavras, caso a inclusão de $closest$ no cluster continue garantindo as restrições, $closest$ é definitivamente (o que significa que esta ação não será mais revertida) adicionado ao cluster e marcado como visitado (linhas 25 a 28).

Quando todos os vizinhos de p tiverem sido avaliados (linha 30), p é removido do $MaxonfootGraph$ (linha 31), e o cluster Lc é adicionado à lista de resultado $stops$, associado ao baricentro ajustado ao mapa s, indicando o local da parada de ônibus (linha 32).

Esse processo é repetido até que todos os pontos tenha sido visitados (linha 34).

Observe que as funções $barycenterAdjust$ e $dMap$ utilizam mapas diferentes. A primeira utiliza um mapa de vias acessíveis de carro, pois ajustam o ponto de parada, enquanto a segunda utiliza um mapa de vias acessíveis à pé, pois é o caminho, à pé, da residência do passageiro até a parada.

Como mostrado na Subseção 2.1, neste trabalho não foi feita manipulação direta dos mapas do OpenStreetMap. Para a manipulação de mapas foi utilizada a ferramenta GraphHopper, que utiliza uma heurística A* para o cômputo de distâncias e provê uma API para o ajuste de pontos ao mapa e manipulação dos mapas OpenStreetMap.

4. Resultados e discussões

O problema do agrupamento para se encontrar o ponto de parada é da classe NP-Hard [Zhang 2018], o que motiva a utilização de diversas heurísticas e meta-heurísticas para

solucioná-lo.

A solução apresentada neste trabalho produz resultados em tempo polinomial, não garante o ótimo global e não foi possível demonstrar o quão próximo do ótimo a solução obtida encontra-se. No entanto, por uma análise qualitativa dos resultados obtidos com dados reais, avaliamos que a solução é muito boa e suficiente do ponto de vista mercadológico.

Os resultados são apresentados nas subseções a seguir através de uma análise assintótica e alguns experimentos. Não foi encontrado na literatura uma solução que se aproximasse da apresentada neste artigo, onde os pontos de paradas podem ser posicionados em qualquer lugar no mapa, característica que motivou o $MaxonfootGraph$, e a aproximação discreta fosse realizada. Por esse motivo não foi possível fazer comparações com outras soluções.

4.1. Análise assintótica

Na Seção 3 foram apresentadas as análises de complexidade dos Algoritmos 1, 2 e 3. Já o Algoritmo 4 é apresentando nesta subseção, pois trata-se da solução principal que de fato resolve o problema de BSS e faz as chamadas dos métodos apresentados anteriormente. A Tabela 2 lista os métodos usados no Algoritmo 4, em que N é o tamanho da lista de estudantes a serem alocados aos pontos de ônibus que serão criados e M é o total de arestas no grafo do mapa que representa o retângulo de obervação, dado pela figura 5.

Tabela 2. Complexidade das funções utilizadas.

Método	Complexidade
discretize	$\mathcal{O}(N)$
discreteSort	$\mathcal{O}(N+K)$
getMaxonfootGraph	$\mathcal{O}(N^2)$
random	$\mathcal{O}(1)$
<list>.peek	$\mathcal{O}(1)$
<list>.remove	$\mathcal{O}(N)$
maxOnfootGraph.sortedNeighbors	$\mathcal{O}(NlogN)$
<list>.append	$\mathcal{O}(1)$
barycenter	$\mathcal{O}(N)$
barycenterAdjust	$\mathcal{O}(1)$
dMap	$\mathcal{O}(M)$

Com as informações referentes às complexidades das funções, é importante a análise das duas estruturas de repetição presentes no Algoritmo 4. A primeira estrutura (linha 7 - 34 do algoritmo) executará até que todas as listas de pontos estejam vazias, essa condição é diretamente relacionada a quantidade de clusters gerado pelo algoritmo, desta forma para uma execução em que foram criados quatro clusters e consequentemente quatro pontos de ônibus, a repetição será executada apenas quatro vezes. Entretanto a análise do loop interno (linhas 20 - 30) é mais complexa e exige que o problema seja dividido nos seguintes casos extremos:

1. Todos os alunos estarão próximos o suficiente, deste modo no $MaxonfootGraph$ todos os vértices estarão associados, apenas um cluster local existirá e por con-

sequência, apenas um ponto de ônibus será criado para atender todos os estudantes.
2. Considerar que os alunos estarão distantes o suficiente, assim o $MaxOnfootGraph$ consistirá apenas de vértices sem nenhum vizinho e, por consequência, $neighbors$ (linha 14 do Algoritmo 4) será vazio.

Para o primeiro caso é fácil verificar que a repetição mais externa será executada apenas uma vez, já que apenas um cluster será criado, dessa forma os métodos <$list$>.$remove$ (linhas 10 a 13) também só serão realizados uma única vez, o mesmo acontece com $maxOnfootGraph.sortedNeighbors$ (linha 17).

Por outro lado, a lista de vizinhos $neighbors$ será do tamanho da lista inicial de alunos, assim o loop interno executará N vezes, a função $barycenter$ executará $\frac{N}{2}$, e assintoticamente continuará sendo N. Já $barycenterAdjust$, como mostrado pela Tabela 2, terá complexidade constante e $dMap$ terá uma execução de $\frac{N}{2} * M$, assintoticamente ficando $N * M$, já o método $mofGraph.remove$ ocerrerá N vezes.

Com essas informações é possível encontrar a complexidade de pior caso para o Algoritmo 4, como sendo $\mathcal{O}(N^2M)$. Sendo que M representa uma quantidade muito grande de arestas, visto que, no grafo que representa o mapa, os vértices não se encontram apenas nos cruzamentos. Até mesmo uma simples rotatória pode ser representada por dezenas de vértices e arestas, para que seja representada o círculo.

No segundo caso temos a melhor execução do Algoritmo 4. Como consequência de $neighbors$ ser vazio a repetição interna não irá sequer ser executada, assim como o método maxOnfootGraph.sortedNeighbors. A complexidade ficará limitada pela execução da função getMaxonfootGraph (linha 4) ou pela repetição externa (linhas 7 a 34) que acontecerá N vezes, e as funções <list>.remove (linhas 10 a 13) que também serão executados N vezes. Por fim, o limite inferior será dado por $\Omega(N^2)$.

Com a análise assintótica do pior e melhor caso é possível perceber que o algoritmo tem complexidade polinomial.

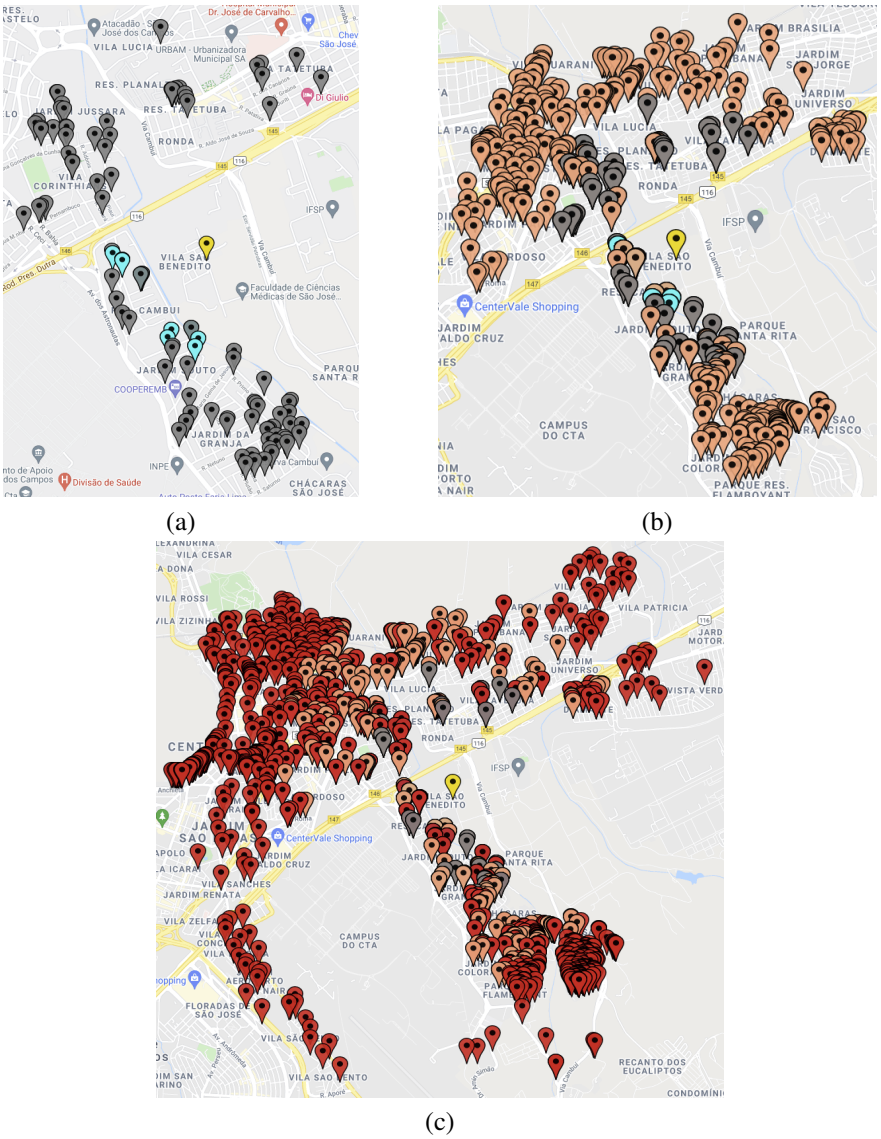
4.2. Experimentos

Os experimentos foram realizados com dados reais de estudantes da rede pública estadual de São Paulo. Os dados foram fornecidos pela Secretaria da Educação do Estado de São Paulo, anonimizados, contendo informações de latitude e longitude da residência de todos os estudantes da rede estadual do município da cidade de São José dos Campos / SP.

No total foram fornecidas as localizações de 11.422 estudantes. Desses, 3.129 pontos (27,4%) estavam localizados em zona rural distante, isolados dos demais ou até mesmo em regiões não mapeadas pelo OpenStreetMap, e foram descartados. O número final de estudantes considerados para este experimento foi de 8.293. Já para os valores dos $maxonfoot$ foram assumidos 300 metros para cada um dos alunos.

Foram gerados cinco cenários de testes a partir do total de pontos. Para gerá-los, foi calculado o centro de massa de todos os 8.293 pontos e os pontos foram ordenados com base na distância até o centro de massa. Assim os cenários foram divididos nas quantidades MINÚSCULA, com os 10 pontos mais próximos do centro de massa, PEQUENA, com os 100 pontos mais próximos, MÉDIA, com 400 pontos, GRANDE, com 1000 pon-

tos e ENORME, com todos os 8.293 pontos. A Figura 7 apresenta uma visualização dos pontos sobre o mapa.



(a) (b)

(c)

Figura 7. Visualização dos cenários de testes. Em (a), o centro de massa (amarelo), o cenário de ordem MINÚSCULA, com 10 pontos (azul) e PEQUENA, com 100 pontos (cinza). Em (b) foi acrescentada o cenário com quantidade MÉDIA, com 400 pontos (salmão) e em (c) foi acrescentada o de 1000 pontos (vermelho).

Uma visualização de todos os 8.293 pontos é mostrada na Figura 8. Observe que muitos pontos chegam a ser localizados em cidades vizinhas. Alguns dos pontos descartados estavam localizados em municípios muito distantes da cidade em estudo.

Ao executar uma única vez o Algoritmo 4 para cada cenário, foram obtidos os resultados apresentados na Tabela 3.

A Figura 9 ilustra os resultados obtidos, com um zoom sobre uma área do mapa apresentando os pontos de parada obtidos para o cenário de quantidade MÉDIA, de 400 pontos. Observe que as paradas são pontos ajustados no mapa. Todas as residências estão

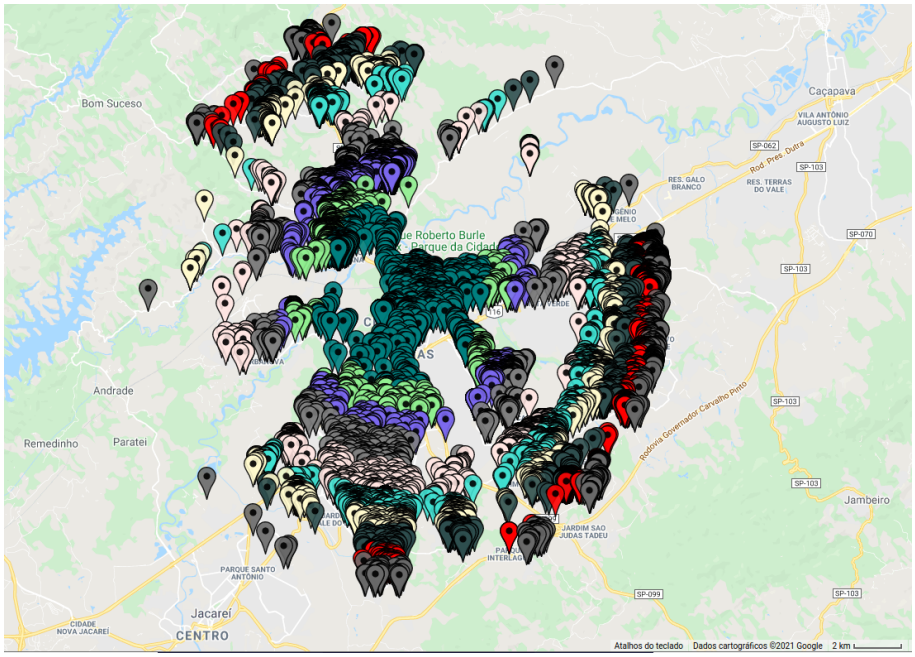


Figura 8. Distribuição dos 8293 pontos sobre o mapa. Cada subconjunto de 1000 pontos está em uma cor diferente.

Tabela 3. Resultados dos experimentos com uma única iteração.

Tamanho	N^o de Pontos	N^o de Paradas
MINÚSCULA	10	2
PEQUENA	100	28
MÉDIA	400	103
GRANDE	1000	323
ENORME	8293	1221

localizadas a, no máximo, 300 metros de distância, à pé, sobre o mapa, de pelo menos uma das paradas obtidas na solução.

A Figura 10 mostra, em maior aproximação, 5 clusters obtidos como resultado, e as Figuras 11 e 12 mostram esses mesmos 5 clusters em uma aproximação ainda maior.

Muitas iterações não mudaram o resultado de forma muito relevante para as entradas com 10, 100 e 400 pontos. Para a entrada com 10 pontos não houve melhora. Por se tratar de uma entrada com muito poucos pontos, uma única iteração já obteve o melhor resultado possível para o algoritmo. Para a entrada de 100 pontos a execução de diversas iterações obteve um resultado com 27 paradas, uma a menos do que a primeira iteração (3,57% menor), enquanto para a entrada de 400 pontos o melhor resultado obtido foi com 100 paradas, 3 a menos do que a primeira iteração (2,91% menor).

Para as entradas de 1.000 e 8.293 pontos foram realizadas 5 execuções isoladas, cada uma com um número diferente de iterações. Os resultados obtidos são apresentados na Tabela 4.

Além das melhores soluções encontradas, a Tabela 4 mostra também as piores so-

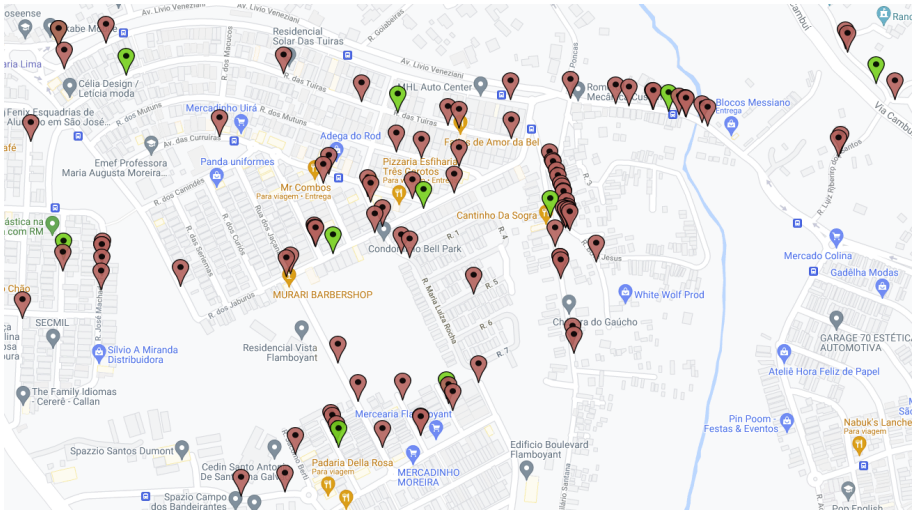
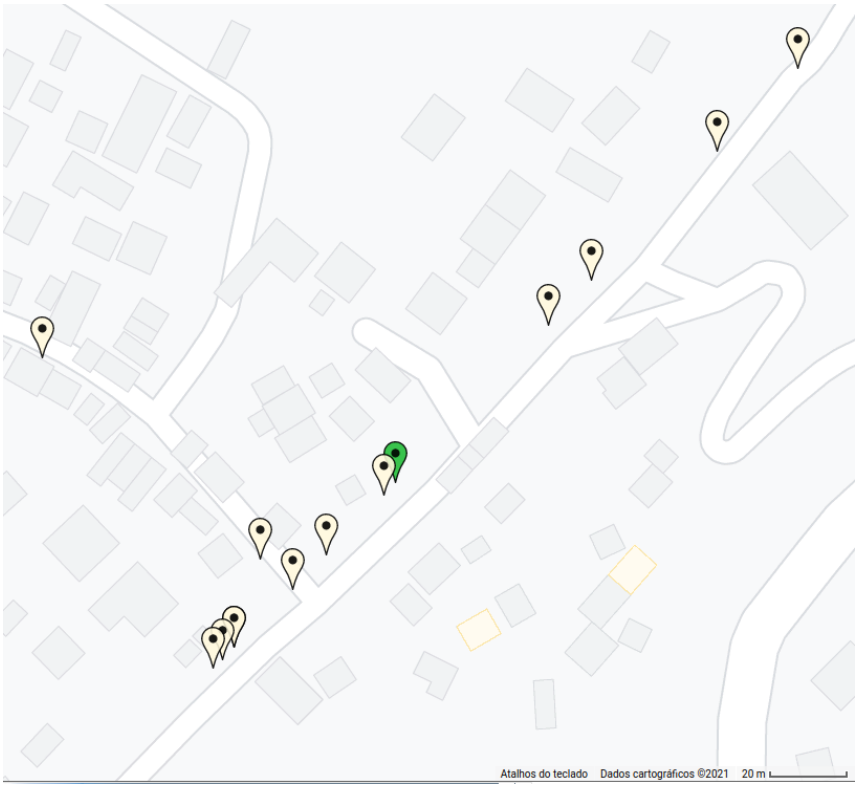
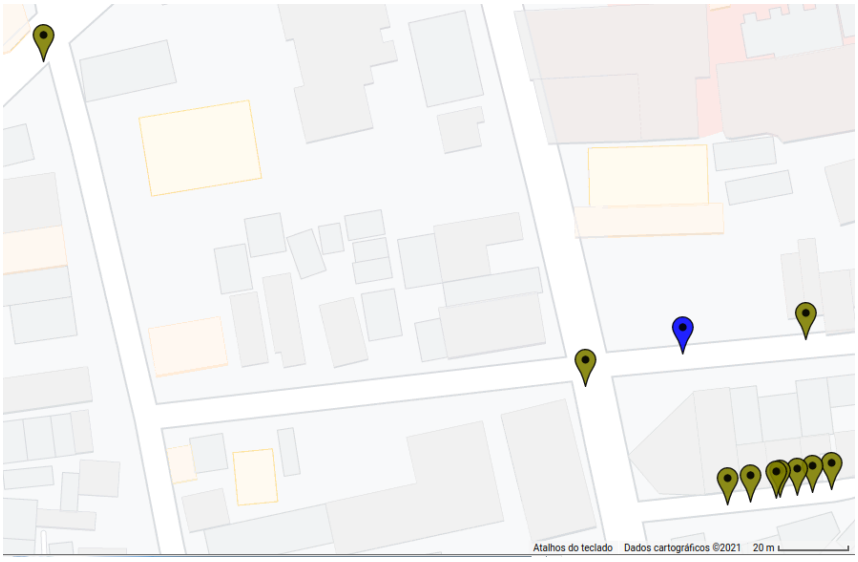


Figura 9. Parte do resultado obtido para a entrada MÉDIA, de 400 pontos. As paradas obtidas são apresentadas em verde.

Figura 10. Resultado incluindo 5 clusters. Cada cluster é representado por uma cor diferente, sendo seu centro representado ainda como uma outra cor.



(a)

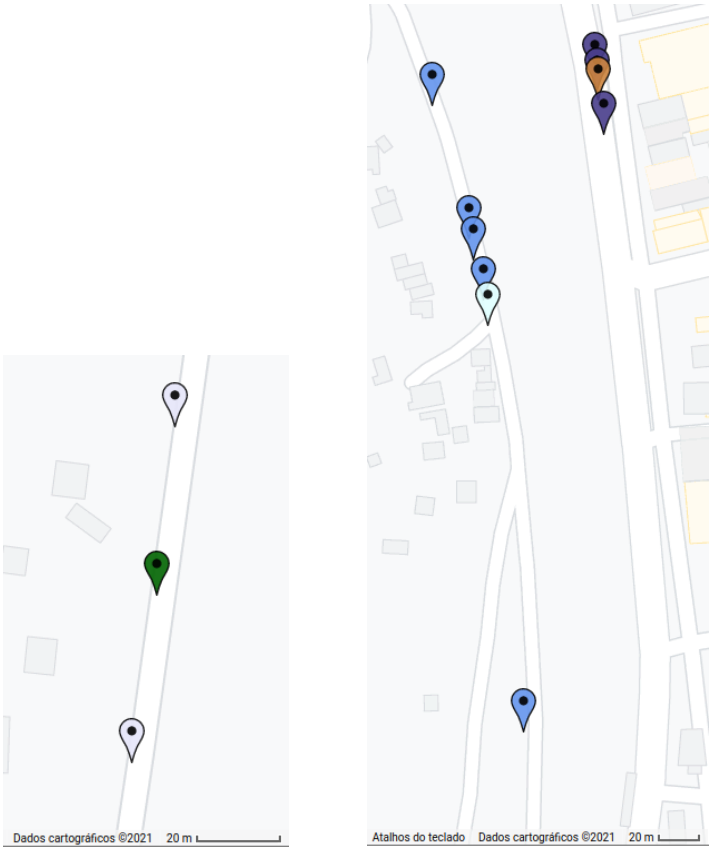


(b)

Figura 11. Zoom sobre dois clusters. (a) parada em verde, compartilhada entre 12 passageiros e (b) parada em roxo, compartilhada entre 10 passageiros.

luções. Isso permite ter uma ideia do quão variável é a solução devido à escolha aleatória da extremidade dos pontos a ser trabalhada a cada iteração do Algoritmo 4. No caso da entrada ENORME, a variação $N_{pior} - N_{melhor}/N_{pior}$ foi de 2,27% e essa variação foi de 2,72% para a entrada GRANDE.

Devido à mesma característica de escolha aleatória, não é possível garantir que



(a) (b)

Figura 12. Zoom sobre três clusters. (a) parada em verde, compartilhada entre 2 passageiros e (b) dois clusters geograficamente próximos, sendo a parada em azul claro compartilhada entre 5 passageiros (azul escuro) e a parada em marrom compartilhada entre 3 passageiros (roxo). Apesar de geograficamente próximos, não há vias de acesso que permita a reunião dos passageiros dos dois clusters de (b) em um única parada.

uma execução com número maior de iterações tenha um resultado melhor do que uma execução com número menor de iterações. Isso pode ser observado nos resultados da entrada de 8.293 pontos apresentados na Tabela 4, pois o melhor resultado foi obtido pela execução com 10 iterações.

É importante ressaltar que um número de iterações tendendo ao infinito garantiria apenas o *melhor resultado possível para o algoritmo*, uma vez que seriam exploradas todas as combinações de todas as iterações do Algoritmo 4. No entanto, **não foi encontrada qualquer relação** entre o melhor resultado possível para o algoritmo e a solução ótima global para o problema (menor número possível de paradas). O algoritmo proposto utiliza uma abordagem gulosa e o máximo que faz é obter uma boa solução em tempo polinomial.

5. Conclusão

Neste artigo foi apresentada uma solução para resolver o problema de seleção dos pontos de paradas de ônibus, que pode ser aplicado a quaisquer conjuntos de dados de pares de coordenadas latitude e longitude que representem localizações de indivíduos que precisem

Tabela 4. Resultados de várias execuções, com diferentes números de iterações. A coluna N_{melhor} mostra o número de paradas da melhor solução encontrada após N_{it} iterações, enquanto a coluna N_{pior} mostra o número de paradas da pior solução encontrada após o mesmo número de iterações. Os melhores e piores resultados obtidos, para todas as iterações, foram destacados em negrito.

Tamanho	Nº de Pontos	N_{it}	N_{melhor}	N_{pior}
GRANDE	1.000	1	323	323
		5	324	328
		10	**322**	329
		20	**322**	328
		100	**322**	**331**
ENORME	8.293	1	1221	1221
		5	1216	1230
		10	**1208**	**1236**
		20	1214	1232
		100	1215	**1236**

ser transportados. Foi possível mostrar que o algoritmo consegue ter um bom desempenho em dados reais, tendo uma complexidade polinomial.

Entretanto, alguns aspectos podem, ainda, ser aprimorados, como a realização de um estudo visando identificar se é possível calcular a aproximação que o algoritmo faz, encontrando um limite de distância entre a solução e o ótimo global (isto é o que define um algoritmo aproximativo).

6. Agradecimentos

Agradecimentos à Fapemig e CNPq pelo financiamento de projetos de pesquisa que permitiram a execução deste trabalho e à Secretaria da Educação do Estado de São Paulo pelo fornecimento dos dados experimentais.

Referências

Bögl, M., Doerner, K. F., and Parragh, S. N. (2015). The school bus routing and scheduling problem with transfers. *Networks*, 65(2):180–203.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, 3rd edition.

David D. Pollard, R. C. F. (2005). *Fundamentals of Structural Geology*. Cambridge University Press.

de Souza, L. V. and Siqueira, P. H. (2010). Heuristic methods applied to the optimization school bus transportation routes: A real case. In García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J. M., and Ali, M., editors, *Trends in Applied Intelligent Systems*, pages 247–256, Berlin, Heidelberg. Springer Berlin Heidelberg.

Desrosiers, J., Ferland, J., Rousseau, J.-M., Lapalme, G., and Chapleau, L. (1981). *An Overview of a School Busing System*, pages 235–243. North-Holland.

Ellegood, W. A., Campbell, J. F., and North, J. (2015). Continuous approximation models for mixed load school bus routing. *Transportation Research Part B: Methodological*, 77:182–198.

Ellegood, W. A., Solomon, S., North, J., and Campbell, J. F. (2020). School bus routing problem: Contemporary trends and research directions. *Omega*, 95:102056.

Faraj, M. F., Sarubbi, J. F., Silva, C. M., Porto, M. F., and Nunes, N. T. R. (2014). A real geographical application for the school bus routing problem. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2762–2767.

Galdi, M. and Thebpanya, P. (2016). Optimizing school bus stop placement in howard county, maryland: A gis-based heuristic approach. *International Journal of Applied Geospatial Research (IJAGR)*, page 15.

GraphHopper (2014). Graphhopper open source. `https://www.graphhopper.com/open-source/`. acessado: 13/06/2021.

Mamajek, E. E., Prsa, A., Torres, G., Harmanec, P., Asplund, M., Bennett, P. D., Capitaine, N., Christensen-Dalsgaard, J., Depagne, E., Folkner, W. M., Haberreiter, M., Hekker, S., Hilton, J. L., Kostov, V., Kurtz, D. W., Laskar, J., Mason, B. D., Milone, E. F., Montgomery, M. M., Richards, M. T., Schou, J., and Stewart, S. G. (2015). Iau 2015 resolution b3 on recommended nominal conversion constants for selected solar and planetary properties.

Martínez, L. M. and Viegas, J. M. (2011). Design and deployment of an innovative school bus service in lisbon. *Procedia - Social and Behavioral Sciences*, 20:120–130. The State of the Art in the European Quantitative Oriented Transportation and Logistics Research – 14th Euro Working Group on Transportation & 26th Mini Euro Conference & 1st European Scientific Conference on Air Transport.

OpenStreetMap (2004). Openstreetmap. `https://www.openstreetmap.org`. acessado: 13/06/2021.

Park, J. and Kim, B.-I. (2010). The school bus routing problem: A review. *European Journal of Operational Research*, 202(2):311–319.

Pérez-Rodríguez, R. and Hernández-Aguirre, A. (2016). Probability model to solve the school bus routing problem with stops selection. *International Journal of Combinatorial Optimization Problems and Informatics*, 7(1):30–39.

Riera-Ledesma, J. and Salazar-González, J.-J. (2012). Solving school bus routing using the multiple vehicle traveling purchaser problem: A branch-and-cut approach. *Computers & Operations Research*, 39(2):391–404.

Riera-Ledesma, J. and Salazar-González, J. J. (2013). A column generation approach for a school bus routing problem with resource constraints. *Computers & Operations Research*, 40(2):566–583.

Sarubbi, J. F. M., Mesquita, C. M. R., Wanner, E. F., Santos, V. F., and Silva, C. M. (2016). A strategy for clustering students minimizing the number of bus stops for solving the school bus routing problem. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 1175–1180.

Worwa, K. (2014). A case study in school transportation logistics. *Research in Logistics & Production*, Vol. 4, No. 1:45–54.

Zeng, W. and Church, R. L. (2009). Finding shortest paths on real road networks: the case for A*.

Zhang, D. (2018). Solving school bus routing and student assignment problems with heuristic and column generation approach.