Roteamento de Frota de Ônibus Escolares

Kevin Wallacy de Souza Maciel[1], **Noé Fernandes**[2]

Instituto Metrópole Digital - Universidade Federal do Rio Grande do Norte (UFRN) - Natal - RN - Brasil

kw_199916@hotmail.com, noepessoa@outlook.com

Abstract

Given that we have a set of student hotspots and a set of heterogeneous capacity school buses. The problem with School Bus Fleet Routing is to be able to route such vehicles to minimize the cost of the school to allocate and service buses, as well as to create routes that minimize the distance from student residences to bus stops. This paper presents a more robust introduction to the problem mentioned earlier (Section 1), then we will present how we can model the problem in graphs (Section 2) as well as how we can solve it through heuristic or exact methods.

Resumo

Dado que temos um conjunto de pontos de concentração de alunos e um conjunto de ônibus escolares de capacidade heterogênea. O Problema de Roteamento de Frota de Ônibus Escolares é conseguir roteirizar tais veículos de forma a minimizar o custo da escola para alocar e prover manutenção aos ônibus, assim como criar rotas que minimizem a distância das residências dos alunos para os pontos de parada dos ônibus. Este documento apresenta uma introdução mais robusta ao problema citado anteriormente (Seção 1), em seguida, vamos apresentar como podemos modelar o problema em grafos (Seção 2), bem como podemos resolvê-lo por meio de métodos exatos ou heurísticos.

1. Introdução

A roterização de veículos sempre foi uma área bastante estudada pela área da logística, pois apresenta um grande desafio matemático (Pois é um problema combinacional NP-Difícil) e um retorno financeiro à altura de tal desafio. O que torna isso bastante interessante no contexto escolar, pois como o ensino é algo bastante custoso, qualquer forma de minimizar gastos sem prejudicar a qualidade do ensino é altamente bem vinda.

Dito isso, algo bastante custoso para os gestores de escolas é justamente o transporte escolar, pois ele acarreta vários custos de manutenção, garagem, e mão de obra para operá-los. Um dos maiores desafios para conceber uma frota eficiente de transporte escolar é estabelecer rotas que favoreçam todos os alunos igualmente, ou seja, que consiga buscar todos os alunos de forma que todos passem o mesmo tempo no transporte (Tanto na ida quanto na volta) e que eles não tenham que se deslocar grandes distâncias para chegar no ponto de coleta mais proximo.

O presente trabalho busca debater sobre os desafios de se modelar esse problema em grafos e sobre as heurísticas que encontramos para tentar reduzir o problema de forma ele consiga ser resolvido em tempo hábil e com o mínimo custo computacional possível.

Este debate está dividido da seguinte forma: Na seção 3 vamos exemplificar o problema real e na seção 3.1 vamos diluir o problema de forma a simplificá-lo afim de utilizar heurísticas já existentes no estado-da-arte. Já na seção 3.2 vamos falar sobre as estuturas de dados que utilizaremos para modelar esse problema. e na seção 3.3 vamos explicar outra forma de visualizar este problema utilizando outro problema similar, que é a roteirização de ônibus urbano. E por fim, na seção 4 vamos apresentar as conclusões que obtivemos ao analizar este problema de roterização e os casos de teste que utilizamos para testar nosso algoritmo.

2. Descrição do Problema Real

A escola Sigma localizada em Natal/RN possui 1000 alunos que precisam todos os dias serem transportados em trajetos de ida e volta da escola para casa. Para atender a essa demanda de alunos, a Sigma possui uma frota de 15 ônibus de capacidades heterogêneas sempre a postos para realizar a tarefa.

Para facilitar o processo de deslocamento dos ônibus e tentar economizar na despesa mensal com a frota, a escola orienta os alunos a se aglomerarem em pontos predeterminados espalhados pela cidade para que os ônibus possam embarcá-los em grandes quantidades. Porém, apesar da ideia fazer sentido, ela não foi muito bem implementada pelo profissional que a executou, de forma que a distância percorrida e quantidade de ônibus utilizada ainda é tão alta quanto anteriormente.

A partir disso, o objetivo do trabalho é fornecer uma solução otimizada para o problema encontrado pela escola citada de forma que sua frota possa atender todos os alunos diminuindo-se a distância percorrida, a quantidade de ônibus utilizada e outros gastos que vêm em consequência desses.

3. Modelagem em Grafos

Para solucionar o problema foi decidido que usaríamos uma instância do famoso Problema do Caixeiro Viajante, uma solução bastante utilizada quando se trata de problemas envolvendo roteamento de veículos. Descreveremos, então, o problema e alguns dos métodos mais conhecidos para sua solução, informando características tais como a complexidade. Logo após, descreveremos detalhadamente o método por nós escolhido para a solução do problema real.

3.1. O Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante consiste em encontrar o menor ciclo hamiltoniano formado sobre um conjunto de vértices. Mais especificamente, o caixeiro parte do vértice inicial, passa por cada vértice uma única vez numa configuração em que o custo seja o menor possível, e retorna para o vértice inicial.

O Caixeiro Viajante é reconhecido como um problema NP-difícil, ou seja, não se conhecem algoritmos ótimos que o resolvam em tempo polinomial. Dessa forma, tem-se que os algoritmos que resolveriam o problema de maneira comprovadamente exata são extremamente custosos, o que traz à tona a necessidade de usar-se heurísticas que possam retornar um resultado aceitável mesmo que nem sempre ótimo.

3.1.1. Soluções exatas

Brute Force

Esta solução consiste em testar todas as permutações possíveis entre os vértices de modo que se possa encontrar a ordem de visitação mais barata. Dentre todas as soluções, a força bruta é a mais simples e também a de pior desempenho, tendo uma complexidade na ordem de O(n!). Esta solução torna-se inviável rapidamente, mesmo que ainda se tenha poucos vértices no grafo.

Passos:

1) Considere o vértice 1 como o inicial e o final.

2) Gere todas as (n - 1)! permutações de vértices.

3) Calcule o custo de cada permutação e observe a de menor custo.

4) Retorne a permutação de menor custo.

Programação Dinâmica

A resolução do caixeiro viajante por programação dinâmica é uma solução bem mais eficiente do que a Brute Force, apesar de ainda não polinomial. Sua complexidade é O($n^2 2^n$). A ideia principal do algoritimo é construir um resultado ótimo resolvendo partes menores do problema e levando os resultados adiante.

Dessa forma, tome o conjunto de vértices 1, 2, 3, ..., n, sendo 1 o primeiro e o último vértice a ser visitado. Então, para cada outro vértice (que não seja 1), nós encontramos o o caminho de menor custo com 1 sendo o vértice inicial e i sendo o vértice final. O custo final desse caminho seria denotado por custo(i) e o custo do ciclo encontrado seria custo(i) + distancia(i, 1). Precisaríamos então encontrar o menor custo dentre todos esses. Para resolver esse problema definimos C(S, i) como sendo o caminho de custo mínimo em que se visita todos os vértices de S exatamente uma vez (começando por 1) e se termina em i.

Passos:

1) Se S tem tamanho 2, o conjunto S deve ser o conjunto {1, i}.

$$C(S, i) = dist(1, i)$$

2) Senão se o tamanho de S é maior do que 2, então

$$C(S, i) = min\{C(S - \{i\}, j) + dist(j, i)\}$$

onde $j \in S, j \neq i$ e $j \neq 1$.

3.1.2. Soluções não-exatas (heurísticas)

Vizinho mais próximo

A heurística do vizinho mais próximo consiste em escolher um vértice inicial arbitrário, visitar o próximo vértice cuja aresta tenha o menor custo até que todos os

vértices sejam visitados. O algoritmo normalmente não atinge um resultado ótimo por sua característica gulosa, ignorando possivelmente caminhos existentes que sejam mais curtos do que o resultado obtido. Mas no geral o algoritmo consegue bons resultados. Sua complexidade é de $O(n^2)$.

Passos:

1) Inicialize todos os vértices como não visitados.

2) Selecione arbitrariamente um vértice. Considere-o como o vértice atual u. Marque u como visitado.

3) Encontre a aresta de menor custo conectando u a um vértice v não visitado.

4) Atribua v a u. Marque-o como visitado.

5) Se todos os vértices do domínio foram visitados, encerre o algoritmo. Senão, vá para o passo 3.

A sequência dos vertices visitados será a saída que o algoritmo obteve.

K-Opt

A ideia da heurística K-Opt é que a partir de um ciclo hamiltoniano inicial se possa realizar periodicamente a substituição de k arestas desse grafo de forma que o custo do comprimento do caminho seja minimizado. Geralmente, as instâncias mais usadas da herística são as 2-Opt e 3-Opt. Finalmente, temos que a complexidade computacional do K-Opt é $O(n^k)$.

Shift

A ideia da heurística Shift é que, dado um ciclo hamiltoniano inicial, realizamos a permutação de um vértice no caminho. Por exemplo, digamos que o caminho hamiltoniano que forma nosso ciclo é descrito como 1-2-3-4-5-6, podemos permutar o vértice 1 da seguinte forma: 2-1-3-4-5-6, assim, podemos percorrer o ciclo seguindo esse caminho (utilzando as arestas de menor custo para seguir a sequência de vértices) e verificar se a permutação melhorou o custo do caminho, caso tenha, armazenamos o resultado e continuamos a tentar todas as permutações para cada vértice do nosso ciclo.

Essa procedimanto é de complexidade temporal $O(n^2)$, mas ele pode ser otimizado com estruturas de dados especializadas ou cuidados especiais para escolher a permutação de vértices.

Passos:

1) Começamos com um ciclo viável no grafo.

2) Salvamos o peso total deste primeiro ciclo e assumimos que ele é a melhor solução.

3) Para cada vértice do ciclo, permutamos ele na sequência afim de encontrar um ciclo de custo menor que o atual.

4) Caso um ciclo de melhor custo seja encontrado, guardamos essa configuração e continuamos com o passo 3 até que tenhamos percorrido todos os vértices.

5) Depois de percorrer todos os vértices, retornamos a melhor solução encontrada.

Exchange

A heurística Exchange consiste em um procedimento em que, tendo-se um ciclo, pode-se trocar a posição de um elemento k por todas as outras posições no ciclo (representado como um vetor) de forma que se possa avaliar a configuração de menor custo. Tendo em vista essa característica, fica claro que a complexidade do algorítimo é de O(n).

Passos:

1) Encontra-se um ciclo com n vértices e define-se uma posição de troca k.

2) Troca-se k por todos as outras posições de vértices do vetor em cada iteração, testando-se se um ciclo de menor custo é atingido.

3) Se alguma troca (melhora) foi realizada, retorna-se o vetor final.

3.2. O Problema do Roteamento de Veículos

O problema do roteamento de veículos busca encontrar uma configuração logística que permita o atendimento de um número V de clientes utilizando N veículos partindo de um ou mais depósitos e retornando aos seus pontos de partida. O objetivo do problema, além de formar rotas disjuntas (a não ser pelo depósito), é diminuir o distância percorrida por cada ônibus, o tempo de viagem e outros custos associados ao percurso realizado, sempre respeitando as restrições de cada tipo de roteamento.

3.2.1. Heurística Clarke & Wright

Dentre todos as mais variados formas de solução do problema do roteamento, escolhemos a heurística Clarke & Wright, que lida com o roteamento de veículos capacitados. A heurística tem característica construtiva, gulosa e baseada em economias. Ou seja, para cada par de vértices deve-se calcular a economia associada a ele da seguinte forma

$$S_{ij} = d_{Hi} + d_{Hj} - d_{ij}.$$

Agora, tendo essas informações em mãos, é possível criar a lista de economias, que se torna a peça chave do algoritmo apresentado a seguir.

INÍCIO
Ler $G = (N,A)$, c_{ij} . {*nó 1 é o depósito central do roteamento*}
Inicializar *Rota*: = ($x_1 - x_s - x_{1)}$,
Calcular a economia s_{ij}: = $c_{1j} - c_{ij} + c_{j1}$ para todo o par de clientes x_i e x_j {*nós em G*}
Ordenar as economias em ordem não crescente e colocá-las em uma lista
Enquanto existir ligações na lista **Faça**
Iniciando pela maior economia da lista **Faça**
Início
Determine a primeira ligação na lista que pode ser utilizada para ser acrescida em um dos dois extremos de ***Rota***, aumentando seu comprimento e a retirando da lista;
Se ***Rota*** não pode ser expandida da forma anterior **então escolha** a primeira ligação na lista para iniciar uma nova rota, e a retire da lista.
Fim
FIM

É importante informar que o algoritmo tem a característica de agrupar o máximo de clientes possível na primeira rota. Além disso, ele também possui a capacidade de receber mais de um tipo de restrição, não apenas de capacidade. Uma outra restrição comumente usada, por exemplo, é a de tempo.

3.3. Análise do problema na literatura

Afim de solucionar o problema fomos em busca de problemas similares na literatura e encontramos dois principais documentos que falavam sobre o tema, os trabalhos de ***R.Bowerman, B. Hall e P. Calami*** e ***R. G. T. e Cláudio Barbieri da Cunha*** (ambos referenciados na seção 5).

O primeiro fala diretamente sobre o problema de rotas de ônibus escolares, ele explica sobre todos os desafios como tamanho do ônibus, preço de manutenção dos veículos, distância que os alunos precisam se locomover para a parada e outras coisas importantes que se fazem necessárias para fazer o devido custeio da implementação da frota.

Mas como os próprios autores explicam, considerar todos esses potenciais problemas tornaria impossível de se encontrar uma solução satisfatória em tempo hábil considerando os algoritmos e heurísticas que possuímos na literatura atualmente. Por isso, em sua solução, eles simplificaram o problema para que assim pudessem chegar o mais próximo de uma solução real e satisfatória.

Primeiro, eles optaram por agrupar os estudantes que moram dentro de um certo raio (que pode variar devido as condições do problema) em uma espécie de cluster e considerar o cluster como sendo um único vértice da rota, este mesmo vértice também representa a parada de ônibus que atende as necessidades daqueles estudantes.

Também foi feita uma simplificação na rota em si, os pontos final e inicial foram definidos como sendo a escola e a noção de que o ônibus sempre visitará as paradas na sua rota e de que não haveriam paradas obsoletas foram definidas para evitar o processamento

excessivo de um mesmo vértice em diversas chamadas do algoritmo.

Com essas simplificações eles obtiveram soluções bastante satisfatórias que não só englobavam todos os alunos de forma justa mas também minimizavam o custo e ciclos da rota, uma vez que não há necessidade do ônibus parar na mesma parada mais de uma vez.

Já o segundo documento aborda o problema de roterização de ônibus urbanos, o que não é exatamente o intuito da nossa pesquisa mas por ser um problema em que podemos aplicar mais simplificações ele acabou contribuindo muito para nossa pesquisa, pois para extender a solução deles para as especificações do ônibus escolar basta inserir os clusters de estudantes como um novo peso para os vértices que já podemos obter resultados satisfatórios e que podem ser utilizados na vida real.

Também tivemos auxílio do livro Otimização Combinatória e Programação Linear: Modelos e Algoritmos, recomendado pela professora. Nele pudemos encontrar mais informações sobre o problema do roteamento de veículos em si e pudemos escolher a heurística Clarke & Wright como método de solução definitivo para o nosso problema.

Assim, após se basear nesses problemas da literatura acreditamos que podemos modelar o problema utilizando os seguintes conjuntos e paramatros.

Conjuntos:

$A =$ Cluster de estudantes organizado por proximidade de caminhada, estes serão os vértices do nosso grafo (ou seja, já consideramos o peso do vértice como sendo o resultado de uma clusterização realizada previamente);

$B =$ Pontos de parada, incluindo a escola, estes serão os vértices que farão parte de uma rota encontrada pelo algoritmo como solução;

$C =$ Conjunto com os ônibus que a escola possui.

Parametros:

$Cap =$ Capacidade de um ônibus.

$Rout =$ Número total de paradas da rota, representado pela quantidade de vértices pertencentes à rota.

3.4. Estrutura de dados

Para representar os grafos, achamos interessante utilizar uma matriz de adjacência e como utilizaremos apenas grafos completos, podemos nos aproveitar e já inserir os pesos das arestas dentro dessa matriz, retirando a necessidade de se criar uma lista adicional para guardar os pesos das arestas. Os pesos dessas arestas será calculado utilizando a distância euclidiana, pois os arquivos que utilizaremos como casos de teste nos fornece um vértice e sua localização geográfica, ou seja, o peso da aresta (i, j) é a distância euclidiana entre i e j.

Também será necessária a criação de uma lista de economias para a utilização do algoritmo do *Clarke & Wright*, esta terá tamaho M, sendo M o número de aresatas do grafo.

3.5. Casos de teste

Os casos de teste utilizados estão disponiveis na biblioteca pública **CVRPLIB**, que pode ser encontrada clickando **aqui**.

Estamos utilizando as instâncias do conjunto *Set A (Augerat, 1995)*. Estes arquivos nos fornecem a quantidade de vértices do grafo, a posição geográfica (posição no eixo x e no eixo y) desses vértices, a quantidade de veículos disponíveis, a capacidade de cada veículo e o peso de cada vértice, que no nosso caso será a quantidade de estudantes clusterizados para aquela parada.

Fizemos uma pequena modificação para facilitar a leitura da quantidade de veículos disponíveis, mas nada que modificasse as configurações originais do grafo.

Todos os arquivos utilizados como intância de teste estão localizados na pasta **Tests/Instances** do projeto.

3.6. Implementação da solução

Para solucionar o problema de roteamento descrito neste documento, utilizamos a abordagem clássica de roteamento de veículos, considereando os alunos como a carga do veículo. Queríamos utilizar a abordagem do caixeiro com restrição de tempo, mas como o tempo seria calculado a partir da distância entre os vértices, encontrar o caminho com menor distância geográfica é equivalente ao menor tempo também, então optamos por considerar apenas a distância, visto que o resultado seria o mesmo.

Já para os algoritmos, utilizaremos a heurística do *clarke and wright*, que irá buscar fechar rotas respeitando a capacidade dos veículos e a quantidade de ônibus disponíveis.

Como a heurística do *clarke and wright* já separa as rotas, não precisamos tomar cuidados adicionais para validá-las, uma vez que o algoritmo implementado corretamente não gerará rotas inválidas.

3.7. Análise de complexidade

O algoritmo Clarke & Wright é conhecido por possuir um bom desempenho computacional, tendo complexidade O(n^2). Quando começamos a implementá-lo com o objetivo de respeitar o limite de sua capacidade. No nosso algoritmo, a complexidade ficou exatamente definida como n^2, sendo o trecho de código que impacta a complexidade é o armazenamento dos cálculos das economias na matriz de economias. Portanto, acreditamos que respeitamos o limite de complexidade do problema.

4. Experimentos

Utilzando a abordagem descrita na seção passada, implementamos a heurística do *clarke and wright* e utilizamos 27 instâncias da biblioteca CVRPLIB para testar o algoritmo e encontrar as rotas para circular os ônibus.

A descrição das rotas e dos casos de teste podem ser encontrados na pasta **Test**, e dentro dela, os casos de teste ficam localizados em **Intances** e os resultados de cada caso, nas pasta **Results**.

Os tempos de execução para cada caso foram:

N	K	Capacidade	Tempo (s)	Rotas fechadas
32	5	100	0.000981815	4
33	5	100	0.000250219	5
33	6	100	0.000253986	6
34	5	100	0.000290395	5
36	5	100	0.000302405	5
37	5	100	0.000312475s	5
37	6	100	0.000343909	6
38	5	100	0.000347677	5
39	5	100	0.000366365	5
39	6	100	0.000376779	6
44	6	100	0.000499581	6
45	6	100	0.000789073	6
45	7	100	0.000537957	7
46	7	100	0.000813211	6
48	7	100	0.000656171	7
53	7	100	0.00079524	7
54	7	100	0.000810198	7
55	9	100	0.000946977	9
60	9	100	0.00115442	8
61	9	100	0.00119567	9
62	8	100	0.00127814	8
63	9	100	0.00129746	9
64	9	100	0.00130705	9
65	9	100	0.00135723	9
69	9	100	0.00159255	9
80	10	100	0.00225415	9

Onde **N** é a quantidade de vértices do grafo, **K** é a quantidade de veículos disponíveis para a escola.

Os experimentos foram realizados numa máquina com um processador **Intel i5** de quinta geração com **2.6Ghz** de potência e uma memória ram de **13Gb** ddr4.

Com esses experimentos, podemos ver que em alguns casos, a quantidade de vértices não influencia muito no tempo de execução, isso se dá ao fato de que o principal "peso" do algoritmo é ditado pela lista de economias, que depende do peso das arestas do grafo, o que não é um fator que depende exclusivamente da característica de um novo vértice inserido no grafo, e sim da relação dele com os outros nós já presentes no sistema.

Também é interessante notar que há casos em que não precisamos utilizar todos os ônibus, uma vez que obtemos rotas vazias como resposta do algoritmo, o que traz uma economia ainda maior para aquele conjunto de soluções.

5. Conclusão

Ainda que tenhamos tido dificuldades principalmente na busca por um algoritmo para solucionar o problema, acreditamos ter finalizado bem o trabalho. Tivemos, então, que a

distribuição dos trabalhos ficou da seguinte forma:

Noé: Preparação das estruturas de dados e implementação da Heurística.

Kevin: Resolver problemas com a implementação (remover a repetição de vértices) e prepar o algoritmo para os casos de teste (ler e salvar os resultados) assim como executar o programa na máquina de testes para analizar os resultados.

Com os resultados em mãos e com a experiência de modelagem adquirida ao longo do semestre, pudemos ver que esse tipo de problema é bastante interessante, pois podemos utilizar uma modelagem relativamente simples em grafos e ainda assim obter resultados bastantes satisfatórios que ajudariam várias pessoas numa situação real, mesmo que ele fosse solucionado por uma heurística, como fizemos aqui.

Também pudemos ver em primeira mão a rapidez de resposta dos métodos heurísticos, que deram respostas bem satisfatórias em um curtíssimo tempo de execução quando comparados aos métodos exatos, fora a experiência de trabalhar com um problema bem atual e realmente impactante na sociedade, o que motivou ainda mais a estudar as soluções para ele.

6. Referências

[1] R. Bowerman, B. Hall, and P. Calamai. A multi-objective optimization approach to urbanschool bus routing: Formulation and solution method. Transportation Research PartA: Policy and Practice, 29(2):107 – 123, 1995.

[2] R. G. T. e Cláudio Barbieri da Cunha. Heurísticas para o problema de dimensionamento e roteirização de uma frota heterogênea utilizando o algoritmo out-of-kilter. 2002.

[3] L. C. Renz. Um algoritmo para roteirização com restrições de tempos de viagens e de trabalho. 1994.

[4] Goldbarg, Marco Cesar Grafos: conceitos, algoritmos e aplicações / Marco Goldbarg, Elizabeth Goldbarg. - Rio de Janeiro: Elsevier, 2012.

[5] GOLDBARG, M. C.; LUNA, H. P. L. Otimização Combinatória e Programação Linear: Modelos e Algoritmos. Editora CAMPUS, 2. ed., 2005.