Roteamento de veículos elétricos para entregas em áreas urbanas

Frank Laércio da C. S. Júnior
frank.junior.118@ufrn.edu.br
Grafos - Elizabeth Ferreira Gouvea Goldbarg e Silvia Maria Diniz Monteiro Maia

Resumo

O objetivo deste relatório é apresentar a resolução do problema de distribuição de mercadorias enviadas por plataformas com veículos elétricos, com o objetivo de calcular as menores rotas e o menor custo utilizando noções e conceitos aprendidos na disciplina de Grafos. Para a realização deste trabalho, analisou-se a resolução de problemas reais através da aplicação de grafos utilizando o Problema do Roteamento de Veículos Capacitados (PRVC), considerando que o veículo possui um limite máximo em sua capacidade de transporte e também energética.

Palavras-chave: Grafos. Problemas reais. Menor rota. Roteamento de Veículos Capacitados.

1. Introdução

Após a Revolução Industrial a emissão de poluentes vem crescendo exponencialmente em nossa sociedade. Porém, mais da metade das emissões de CO_2 foram feitas nos últimos 30 anos, conforme os dados do *Global Carbon Project*.

No Brasil, segundo dados do Sistema de Estimativas de Emissões de Gases de Efeito Estufa (SEEG), no ano de 2020, a emissão de gases de efeito estufa cresceram 9,5%, mesmo que durante a pandemia o mundo tenha poluído 7% a menos no período de confinamento. Ou seja, nós como sociedade caminhamos para um colapso ambiental e isso nos faz refletir sobre como lidamos com o nosso meio ambiente e o que podemos fazer para ajudá-lo.

Portanto, o objetivo deste relatório é aplicar os conceitos teóricos e práticos aprendidos em sala de aula na disciplina de Grafos para cenários cotidianos e buscar meios de ajudar na diminuição de emissão de gases poluentes e custos na logística de entrega de encomendas realizadas utilizando veículos elétricos.

2. Descrição do problema real

A situação-problema em questão visa otimizar os custos logísticos das operações de envio de encomendas aos clientes, levando em consideração diversos aspectos, como tempo de entrega, custo de envio, recarga da bateria dos veículos elétricos e a menor quantidade de viagens possíveis. Para isso, as empresas possuem uma rede de depósitos estrategicamente distribuídos em zonas específicas, bem como uma frota de veículos disponíveis para realizar as entregas até o destino final.

A fim de alcançar o objetivo de otimização dos recursos, é fundamental maximizar o número de entregas realizadas por cada veículo, partindo de um depósito inicial e percorrendo a menor rota possível para cada entrega. Essa funcionalidade é obtida por meio do uso de algoritmos baseados em grafos, que tratam especificamente de veículos capacitados, levando em consideração a capacidade dos veículos elétricos e as restrições operacionais.

A aplicação de algoritmos em grafo permite modelar o problema de roteamento de veículos elétricos de entrega em áreas urbanas, considerando a localização dos pontos de entrega e recarga como vértices, e as rotas possíveis entre eles como arestas. Com base nessa modelagem, os algoritmos são capazes de encontrar soluções otimizadas que minimizam a distância percorrida pelos veículos, reduzem os custos operacionais e contribuem para uma operação logística eficiente e sustentável.

Portanto, a abordagem proposta busca utilizar algoritmos em grafo para resolver o problema de roteamento de veículos elétricos de entrega em áreas urbanas, visando otimizar a utilização dos recursos disponíveis, minimizar os custos logísticos e proporcionar um serviço de entrega eficiente aos clientes.

3. Modelagem em grafos

A modelagem do grafo é feita com base nos seguintes participantes chaves do problema: cliente, veículo, rota, recarga e depósito. O cliente é descrito no grafo com o vértice e tem a sua capacidade esperada para uma determinada compra. Já com o veículo é simplesmente levado em consideração a sua capacidade para realizar uma viagem. A rota é a otimização feita pelo algoritmo, levando em consideração a capacidade dos veículos e o menor percurso a ser feito, sendo representada visualmente com as arestas. Já os pontos de recargas podem recuperar a capacidade máxima do veículo durante a rota. Por fim, o(s) depósito(s) são representados como capacidade zero(s) e é onde partem os veículos.

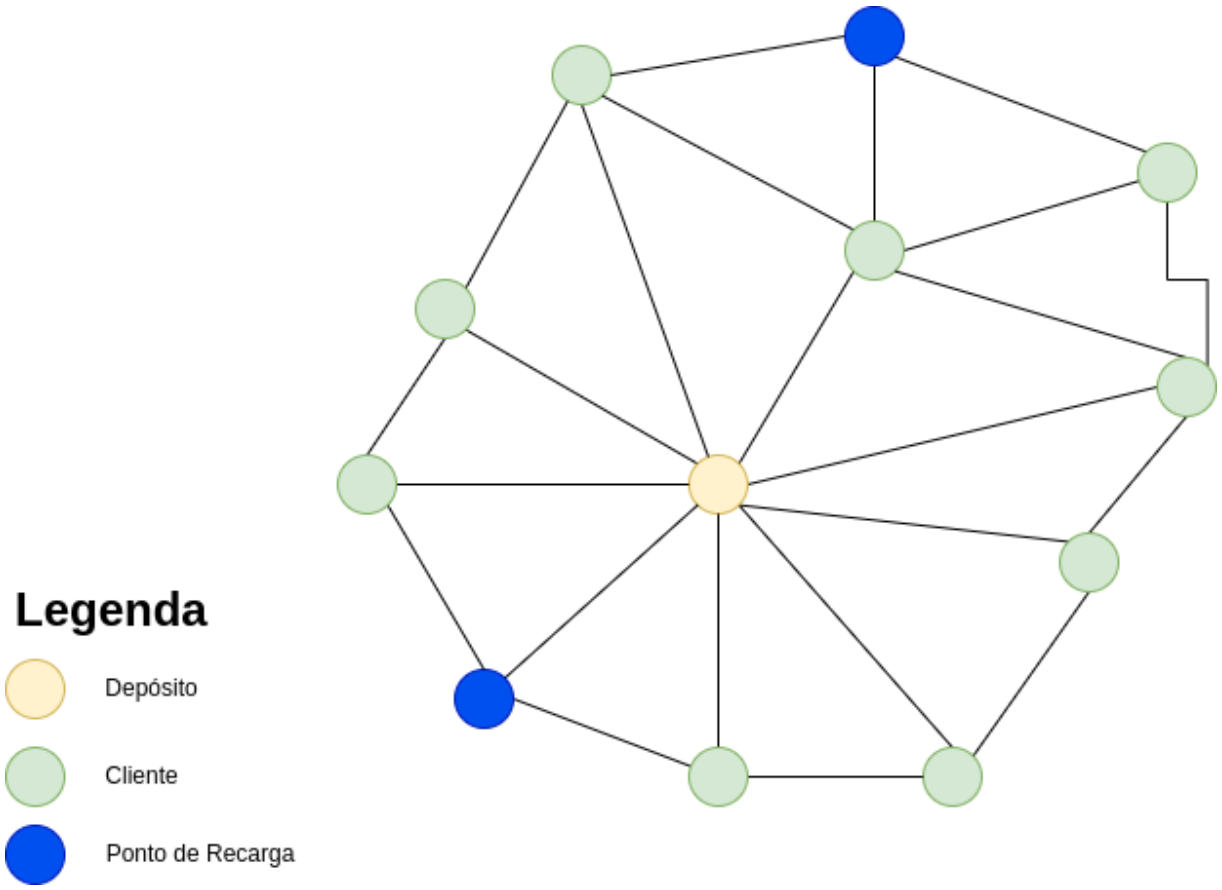

Figura 2 - Esquema gráfico de solução de um Problema de Roteamento de Veículos (PRV), com um só depósito.

Fonte: produção do autor.

Na figura 2 é possível visualizar um exemplo de como é uma determinada região atendida por essa empresa. O nó central (com capacidade zero) é o depósito, e é dele onde partem as rotas realizadas pelos veículos. Além disso, cada vértice representa um cliente que tem uma capacidade definida pelo produto que ele irá receber.

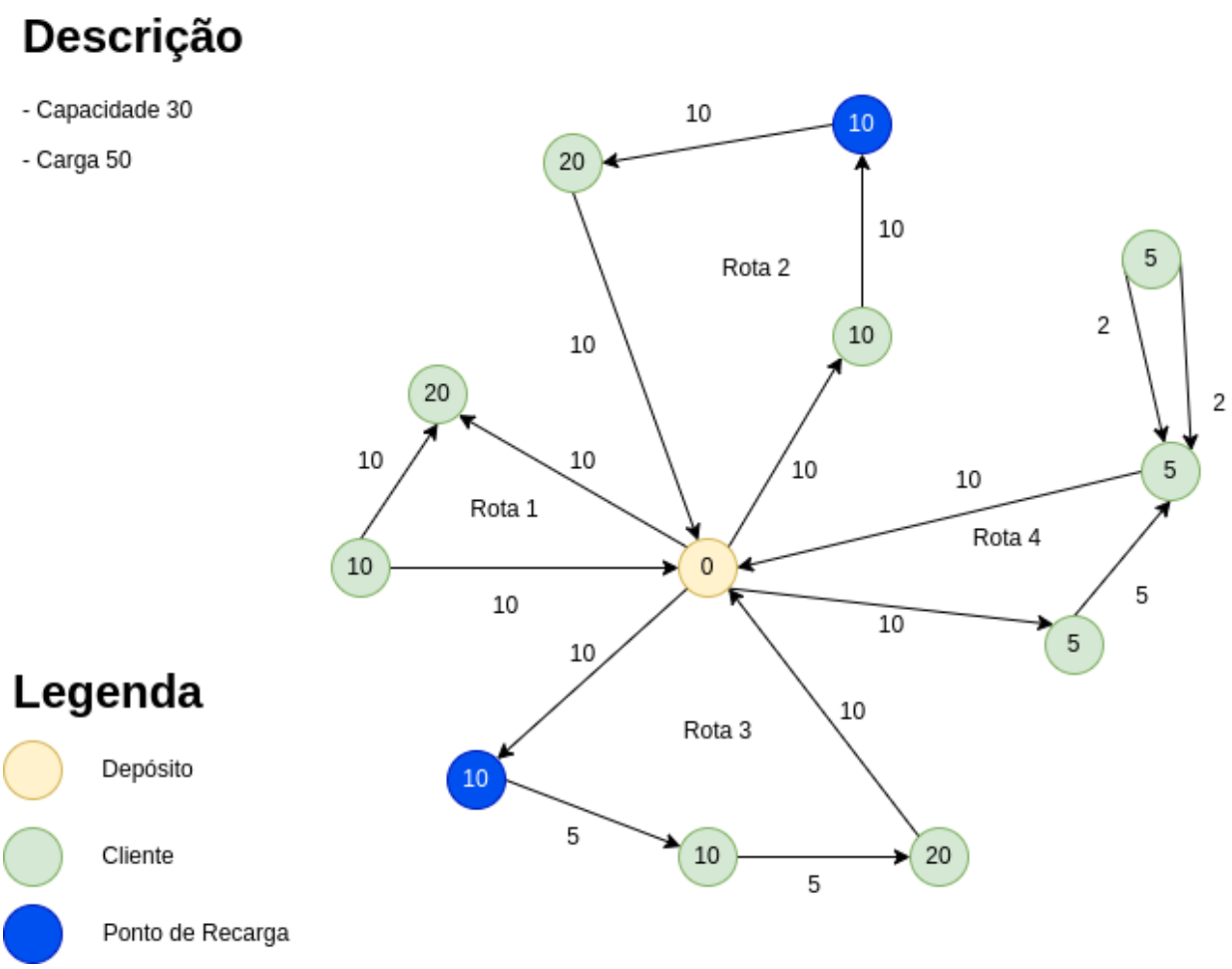

Figura 3 - Exemplo da aplicação do algoritmo.

Fonte: produção do autor.

Por fim, a resolução das entregas dos produtos será executada respeitando todos os seguintes critérios abaixo:

- Deverá partir de um depósito central (Figura 2).
- Os veículos possuem capacidade máxima de 50 para as suas encomendas (Figura 3).
- A autonomia da bateria é de no máximo 100 (Figura 3).

- Cada aresta é ponderada com o custo da distância percorrida (Figura 3).
- Cada cliente deve ter a sua capacidade descrita com base na capacidade da compra realizada.
- As rotas devem ser calculadas utilizando o algoritmo de Clarke Wright (*Saving Routes*).

4. Estado da arte e revisão bibliográfica

O Problema do Roteamento de Veículos Capacitados (PRVC) é um assunto que vem crescendo nas últimas décadas devido à otimização de recursos energéticos por boa parte da sociedade (LEITÃO e SALIM, 2020). Por esse motivo a logística que envolve esse problema é o foco das discussões acadêmicas, empresariais, governamentais e ambientais. Os fatores que influenciam são: a capacidade de carga de cada um veículo de transporte, quantas viagens são necessárias para fazer o abastecimento de todos os destinos, o número disponível de automóveis e o tempo de escoamento também pode ser um dos fatores que afetam a rede de distribuição.

As abordagens para a resolução do problema proposto podem ser divididas em: métodos exatos e métodos heurísticos.

4.1. Métodos exatos

Métodos exatos segundo Leeuewn são soluções que garantem que a solução ótima será encontrada, mas que pode ser custoso computacionalmente devido à complexidade NP-difícil (OLIVEIRA e DELGADO, 2015).

Um algoritmo utilizado para encontrar em uma solução exata é o “Algoritmo de Branch-and-Bound” para resolver o PRVC. Em resumo consiste no encadeamento de todas as soluções possíveis, logo após serem consideradas as restrições, os que forem sendo considerados soluções ruins são descartados e no final do algoritmo restarão somente as soluções otimizadas (Fischetti, Toth e Vigo, 1994).

Além disso, essa solução parte do princípio de se utilizar “upper bound”, ou limites superiores, para descartar as soluções ruins, uma vez que já sabendo que não atende ao critério estabelecido no “upper bound”, ela já é automaticamente descartada.
A complexidade do algoritmo de “Branch-and-Bound” é na ordem exponencial, visto que são analisadas todas as possíveis soluções para resolver o problema. Apesar desse motivo da complexidade, é o ideal quando se busca a solução ótima para resolver o problema do roteamento de veículos.

4.2. Métodos heurísticos

Já os métodos heurísticos “[...] são métodos que percorrem um caminho reduzido no espaço de busca seguindo uma estratégia guiada e não garantem que a solução ótima será encontrada, mas permitem encontrar uma aproximação com menor custo de tempo e recursos” (OLIVEIRA e DELGADO, 2015).
No entanto, o seu desempenho pode ser considerado ruim, pois na fase do plano cortante pode ocorrer um número alto de interações para encontrar essa solução ótimo, tornando o programa linear insolúvel devido ao número elevado da árvore gerada (MIRANDA, 2011).

O algoritmo de Clarke & Wright Savings é um dos heurísticos estudos em PRVC sendo baseado no princípio da economia de recursos (Clarke e Wright, 1964). A ideia é que a cada passo do algoritmo os fatores de restrições são considerados.

O princípio do algoritmo se baseia na ideia de que se há duas rotas diferentes r = (0, ..., i, 0) e s = (0, j, ..., 0) que são factíveis de uma fusão (merge) em uma nova rota t = (0, ..., i, j, ...0), a fusão dessas rotas gera uma economia de custo Sij [10], definida por: Sij = C(0, i) + C(0,j) - C(i,j) (OLIVEIRA e DELGADO, 2015).

Logo, a junção dos caminhos são factíveis quando a soma de um ou mais rotas não excedem a capacidade mais do veículo. Sendo Q a carga máxima de cada automóvel. Para este algoritmo de Clarke e Wright Savings existe a versão paralela e sequencial. A diferença consta que na versão paralela mais de uma rota é considerada na construção do caminho.

5. Casos de teste

Na literatura, é possível encontrar uma variedade de exemplos de casos de teste para problemas de roteamento de veículos elétricos de entrega em áreas urbanas. Esses casos de teste são utilizados para avaliar a eficácia e o desempenho dos algoritmos propostos.

5.1. **Conjunto de pontos de entrega e recarga:** eles podem incluir um conjunto de pontos de entrega e recarga, cada um com suas coordenadas geográficas e demanda associada. Esses pontos podem ser distribuídos em uma área urbana, e o objetivo é encontrar as rotas ótimas para entregar as encomendas aos clientes, levando em consideração a capacidade do veículo elétrico e as restrições operacionais (Li J., Wang F., He Y., 2020).

5.2. **Diferentes capacidades de veículos**: nesses casos de testes podem considerar diferentes capacidades de veículos elétricos. Isso permite avaliar como os algoritmos lidam com capacidades distintas e como isso afeta o número de viagens necessárias e a eficiência geral das rotas (Mohamadi M., Javadian, 2020).

5.3. **Restrições de tempo de entrega**: cada ponto de entrega possui um prazo limite para ser atendido, sendo o objetivo encontrar rotas que garantam que todas as entregas sejam realizadas dentro dos prazos estabelecidos, levando em consideração as limitações de tempo e a capacidade do veículo (Hu QIN, et al, 2021).

5.4. **Comparação com soluções ótimas conhecidas**: Alguns casos de teste podem ser criados com base em instâncias de problemas clássicos já conhecidos na literatura, onde as soluções ótimas já foram calculadas. Isso permite comparar o desempenho dos novos algoritmos propostos com as soluções ótimas conhecidas, avaliando sua eficiência e

capacidade de se aproximar do ótimo global (Li J., Wang F., He Y., 2020).

6. Links

Abaixo encontram-se enumerados os links relacionados ao tema do Problema de Rotas de Veículos Capacitados (PRVC).

6.1. Sistema para roteamento de veículos capacitados aplicando métodos de Monte Carlo. Disponível em: <https://sol.sbc.org.br/index.php/sbsi/article/view/5795>. Acesso em 15 mai. 2022.

6.2. Metaheurísticas para as variantes do problema de roteamento de veículos: capacitado, com janela de tempo e com tempo de viagem estocástico. Disponível em: <https://repositorio.ufmg.br/handle/1843/BUOS-8SJKX4>. Acesso em 15 mai. 2023.

6.3. Problema de roteamento de veículos capacitados (PRVC): solução manual x busca dispersa. Disponível em: <http://ws2.din.uem.br/~ademir/sbpo/sbpo2012/pdf/arq0330.pdf> . Acesso em 15 mai. 2023.

6.4. O Papel Da Logística Reversa Na Mitigação Do Desperdício Em Cadeias De Suprimentos Agroalimentares. Disponível em: <https://saber.unioeste.br/index.php/gepec/article/view/24493>. Acesso em 15 mai. 2023.

6.5. Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations Research, 12(4), 568-581. Disponível em: https://doi.org/10.1287/opre.12.4.568>. . Acesso em 15 mai. 2023.

6.6. Hu QIN, Xinxin SU, Teng REN, Zhixing LUO. Electric vehicle routing problem: A review of models, algorithms and applications. Journal of

Cleaner Production, 208, 925-946. 2019. Disponível em: <https://journal.hep.com.cn/fem/EN/10.1007/s42524-021-0157-1>. Acesso em 16 abr. 2023.

7. Descrição do Algoritmo

Para o projeto será utilizado o algoritmo de Clarke e Wright, também conhecido como *Savings Algorithm (Algoritmo de economia)*, sendo utilizado para resolver o problema de roteamento de veículos. Nele envolve a otimização do uso de uma frota ou de um único veículo para atender a um conjunto de clientes com demandas específicas. O algoritmo foi proposto (G. Clarke e J. W. Wright, 1964) é amplamente utilizado para encontrar rotas eficientes e econômicas para a entrega de mercadorias, coleta de resíduos, transporte escolar, entre outros cenários do cotidiano.

A ideia principal do algoritmo é identificar pares de clientes que possam compartilhar parte de uma rota, de modo a reduzir o custo total das viagens. O algoritmo segue os seguintes passos:

1. Calcular a economia (savings) entre cada par de clientes. A economia é definida como a diferença entre o custo de atender cada cliente individualmente e o custo de atender ambos os clientes em uma única rota. Essa economia pode ser calculada usando uma métrica de distância, tempo de viagem ou gasto de combustível.
2. Ordenar as economias em ordem decrescente, do maior para o menor (Para isso será utilizado o MergeSort (Cormen H. T., 2009) para ordenação).
3. Inicialmente, cada cliente é atribuído a uma rota separada.
4. Para cada par de clientes com a maior economia, verifica se é possível combinar suas rotas sem violar as restrições de capacidade dos veículos. Caso seja possível, os clientes são colocados na mesma rota, compartilhando uma parte comum.

5. O passo anterior é repetido para os próximos pares de clientes com economias decrescentes, até que não seja possível mais combinar rotas ou não haja mais economias positivas.

6. No final, é obtida uma solução com rotas otimizadas que compartilham trajetos e reduzem o custo total.

O algoritmo de Clarke e Wright é relativamente simples e eficiente, especialmente para instâncias de VRP pequenas a moderadas. No entanto, para problemas com restrições mais complexas, como janelas de tempo, múltiplos depósitos ou restrições de precedência, podem ser necessários algoritmos mais avançados para obter soluções ótimas.

O algoritmo foi implementado utilizando a linguagem de programação Java na versão estável 17 (LTE). Vale salientar que não foi utilizada nenhuma biblioteca para o desenvolvimento do algoritmo, além dos próprios mecanismos que a linguagem proporciona.

Seguindo a ideia central o algoritmo começa lendo um arquivo com uma base de dados gerada de forma aleatória sendo a primeira linha representando o depósito e as demais os clientes atendidos. As colunas representam os seguintes dados: id (identificador do cliente ou depósito), custo da demanda do cliente, posição no eixo x e posição no eixo y (para o cálculo da distância).

```
public static void main(String[] args) {
  List<Customer> customers = FileUtil.readCustomersFromCSV("data/test1.csv");
```

Figura 4 - Código de importação das massas de testes.

Fonte: produção do autor.

Após a importação (Figura 4) é instanciada a classe que irá calcular as rotas seguindo o algoritmo de Clarke e Wrights levando em consideração a carga do veículo

(*load*) e a capacidade da bateria (*batteryCapacity*) para calcular a capacidade total que o veículo pode aguentar.

```
public static void main(String[] args) {
  List<Customer> customers = FileUtil.readCustomersFromCSV("data/test1.csv");

  ClarkeWrightAlgorithm clarkeWrightAlgorithm = new ClarkeWrightAlgorithm();

  int load = 2; // 2 is the load of each vehicle in tons
  int batteryCapacity = 100; // 100 is the battery capacity in kWh
  int vehicleCapacity =  batteryCapacity / load;

  List<Route> routes = clarkeWrightAlgorithm.solve(customers, vehicleCapacity);
```

Figura 5 - Código que insere a capacidade do veículo.

Fonte: produção do autor.

Após as configurações iniciais (Figura 5) o algoritmo chama o método *solve* para resolver o problema utilizando a abordagem clássica de Clarke & Wright. Vale salientar que a modelagem dos dados foram feitas utilizando as seguintes classes.

- **Customer** [id (identificador), demand (demanda), x (distância em x), y (distância em y)]
- **Node** [from (cliente de origem), to (cliente de destino), savings (custo)]
- **Route** [customers (lista de clientes atendidos)]

Logo, para calcular as rotas seguindo a capacidade do veículo foi utilizado o método solve recebendo a lista de clientes e a capacidade do veículo. Após isso foi seguido os passos de calcular as economias (savings), ordenar a lista de clientes de maior custo para de menor custo e instanciar as rotas partindo do depósito.

```
public List<Route> solve(List<Customer> customers, int vehicleCapacity) {
  List<Node> savings = calculateSavings(customers);
  MergeSort.sort(savings);
  List<Route> routes = constructInitialRoutes(customers);
```

Figura 6 - Código que calcula os custos, ordena e inicia as rotas

Fonte: produção do autor.

O cálculo das economias é feito utilizando a distância do cliente A até o depósito, somado à distância do cliente B até o depósito e subtraído da distância de A e B. Isso representa a parte do algoritmo que calcula o custo de cada ponto sair do depósito, atender um cliente e voltar.

```
double saving = (customerA.distance(customers.get(depotIndex)))
    + (customerB.distance(customers.get(depotIndex)))
    - (customerA.distance(customerB));
```

Figura 7 - Cálculo do custo de ir e voltar do depósito seguindo o algoritmo de Clarke & Wright

Fonte: produção do autor.

Por fim, é feita a junção das rotas respeitando a capacidade do veículo conforme a demanda do ponto A, somada à demanda do ponto B não ser maior do que a capacidade do veículo.

```
for (Node saving : savings) {
  Customer customerA = saving.getFrom();
  Customer customerB = saving.getTo();
  Route routeA = findRouteContainingCustomer(routes, customerA);
  Route routeB = findRouteContainingCustomer(routes, customerB);

  if (routeA ≠ null && routeB ≠ null && routeA ≠ routeB
      && (routeA.getDemand() + routeB.getDemand() ≤ vehicleCapacity)) {
      Route mergedRoute = mergeRoutes(routeA, routeB);
      routes.remove(routeA);
      routes.remove(routeB);
      routes.add(mergedRoute);
  }
}
```

Figura 8 - Junção das rotas seguindo o algoritmo de Clarke & Wright

Fonte: produção do autor.

O método que faz a junção das rotas (mergeRoutes) da Figura 8 somente adiciona na rota os clientes de A e B.

8. Descrição dos experimentos computacionais

Os experimentos foram realizados na máquina pessoal do autor que possui as seguintes configurações conforme o quadro 1.

Quadro 1 - Componentes utilizados nos testes

Componente	**Peça**
Processador	Intel i5 10th (6 núcleos e 12 Threads)
Memória	16Gb de RAM DDR4 3200Mhz
Disco	500Gb SSD Nvme
Placa de vídeo	GTX 1050Ti
Sistema Operacional	PopOs 22.04

Fonte - produção do autor

Para os testes foi considerado, para simplificação do projeto, que a capacidade do veículo leva em consideração a carga da bateria para atender aos clientes. Logo, foram utilizados 2 cenários de testes, sendo a primeira linha representando o depósito.

Quadro 2 - Cenário 1 de testes

identificador	**demanda**	**x**	**y**
0	0	0	0
1	10	5	13
2	18	7	19
3	23	18	11
4	32	11	7
5	15	9	4
6	41	14	16
7	8	6	2

8	28	3	9
9	12	16	5

Quadro 3 - Cenário 2 de testes

identificador	demanda	x	y
0	0	0	0
1	17	8	10
2	29	12	4
3	7	3	6
4	20	14	18
5	42	7	11
6	13	2	16
7	35	17	9
8	9	5	3
9	26	10	15

Fonte - produção do autor

9. Resultados obtidos

Os resultados apresentados abaixo mostram que foi possível atingir o objetivo de economia de recursos respeitando a capacidade dos veículos.

Quadro 4 - Resultados obtidos

Cenário de teste	**Resultados**
Cenário 1	Route 0: 0 (0.0, 0.0) 6 (9.0, 16.0) 0 (0.0, 0.0) Total distance: 36.71511950137164 Total demand: 41

	Route 1: 0 (0.0, 0.0) 8 (6.0, 9.0) 0 (0.0, 0.0) Total distance: 21.633307652783937 Total demand: 28 Route 2: 0 (0.0, 0.0) 3 (2.0, 11.0) 5 (11.0, 4.0) 0 (0.0, 0.0) Total distance: 34.28679404920995 Total demand: 38 Route 3: 0 (0.0, 0.0) 4 (18.0, 7.0) 7 (14.0, 2.0) 0 (0.0, 0.0) Total distance: 39.858467776991766 Total demand: 40 Route 4: 0 (0.0, 0.0) 1 (5.0, 13.0) 2 (7.0, 19.0) 9 (3.0, 5.0) 0 (0.0, 0.0) Total distance: 40.644115270927216 Total demand: 40 Process finished with exit code 0
Cenário 2	Route 0: 0 (0.0, 0.0) 5 (7.0, 11.0) 0 (0.0, 0.0) Total distance: 26.076809620810597 Total demand: 42

	Route 1: 0 (0.0, 0.0) 7 (17.0, 9.0) 0 (0.0, 0.0) Total distance: 38.47076812334269 Total demand: 35 Route 2: 0 (0.0, 0.0) 9 (10.0, 15.0) 0 (0.0, 0.0) Total distance: 36.05551275463989 Total demand: 26 Route 3: 0 (0.0, 0.0) 1 (8.0, 10.0) 2 (12.0, 4.0) 0 (0.0, 0.0) Total distance: 32.66646166646719 Total demand: 46 Route 4: 0 (0.0, 0.0) 3 (3.0, 6.0) 4 (14.0, 18.0) 6 (2.0, 16.0) 8 (5.0, 3.0) 0 (0.0, 0.0) Total distance: 54.32516554816715 Total demand: 49 Process finished with exit code 0

Fonte - produção do autor

Para executar o programa e testar deve-se selecionar o caso de teste e executar a função main para obter os resultados, vale salientar que pode variar de acordo com máquina que está sendo utilizada, mas não o resultado em si que foi apresentado.

10. Conclusão

O problema de roteamento de veículos elétricos de entrega em áreas urbanas é um desafio complexo que envolve a otimização de recursos levando em consideração restrições de autonomia da bateria, disponibilidade de pontos de recarga, restrições de tempo de entrega e de entregas.

Através da aplicação de técnicas de otimização, como o algoritmo de Clarke & Wright modificado, é possível encontrar soluções eficientes que reduzem a distância percorrida pelos veículos elétricos e, assim, economizam energia e reduzem as emissões de poluentes.

Ao utilizar uma abordagem baseada em grafos e considerar a autonomia dos veículos e a disponibilidade de pontos de recarga ao longo das rotas, é possível obter um roteamento otimizado que contribui para a eficiência e sustentabilidade das operações de entrega em áreas urbanas.

Essa solução traz benefícios tanto para as empresas de logística, que podem reduzir custos operacionais e melhorar a eficiência de suas operações, quanto para o meio ambiente, ao promover a utilização de veículos elétricos de forma mais sustentável. Portanto, investir em técnicas de otimização do roteamento de veículos elétricos de entrega em áreas urbanas é fundamental para alcançar um transporte mais eficiente, sustentável e alinhado com as necessidades das cidades modernas.

Referências bibliográficas

[1] DE CARVALHO OLIVEIRA, R. A.; DELGADO, K. V.; MOREIRA, D. A. Sistema para roteamento de veículos capacitados aplicando Métodos de Monte Carlo. iSys - Brazilian Journal of Information Systems, [S. l.], v. 8, n. 3, p. 42–63, 2015. DOI: 10.5753/isys.2015.290. Disponível em: https://sol.sbc.org.br/journals/index.php/isys/article/view/290. Acesso em: 02 abr. 2023.

[2] Marco Goldbarg e Elizabeth Goldbarg. Grafos. Conceitos, algoritmos e aplicações. 7ª tiragem. São Paulo: Elsevier Editora Ltda. 2012. 622 páginas.

[3] Renato Ferreira de Souza. Resolução de problemas via teoria de grafos. São Carlos. 2014.

[4] G. Clarke and J. W. Wright. Scheduling of vehiclesfrom a central depot to a number of delivery points.Operations research, 12(4):568–581, 1964.

[5] Araujo C. Problema de Veículos Capacitados. Disponível em: <https://upload.wikimedia.org/wikipedia/commons/3/30/PRVC.jpg>. Acesso em 02 abr. 2023.

[6] Li, X., et al.. A survey of electric vehicle routing and charging optimization: Models, algorithms, and applications. 2020. Energies, 13(4), 873. Disponível em: <https://www.mdpi.com/2071-1050/12/24/10537>. Acesso em 16 abr. 2023.

[7] Murcia, C. L. D., Proudhon C., Afsar M. H.. The electric vehicle routing problem with time windows, partial recharges and satellite customers. 2020. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S1366554518310597>. Acesso em 16 abr. 2023.

[8] Hu QIN, Xinxin SU, Teng REN, Zhixing LUO. Electric vehicle routing problem: A review of models, algorithms and applications. Journal of Cleaner Production, 208, 925-946. 2019. Disponível em: <https://journal.hep.com.cn/fem/EN/10.1007/s42524-021-0157-1>. Acesso em 16 abr. 2023.

[9] Cormen H. T., et. al. Introduction to Algorithms, Third Edition. The MIT Press. 2009.