UNIVERSIDADE FEDERAL DE ITAJUBÁ

SMAC03 - GRAFOS

COLETAS DE DADOS IMOBILIÁRIOS

2020026106 - ANDRE MARCOS FERREIRA
2021001881 - JUAN PABLO RIBEIRO
2023008139 - MARIA EDUARDA RIBEIRO
2022008608 - PEDRO HENRIQUE BORGES DA SILVA

SMAC03 - GRAFOS

Prof. Rafael Frinhani

INSTITUTO DE MATEMÁTICA E COMPUTAÇÃO
UNIFEI - Itajubá

2025

UNIVERSIDADE FEDERAL DE ITAJUBÁ
INSTITUTO DE MATEMÁTICA E COMPUTAÇÃO
SISTEMAS DE INFORMAÇÃO
SMAC03 - GRAFOS

Coletas de Dados Imobiliários

1 Introdução

Este relatório visa a abordagem com relação ao estudo feito acerca do tema de Coletas de Dados Imobiliários. O objetivo principal do estudo foi feito visando melhorar o planejamento de coleta e reduzir o esforço e o tempo para essa demanda.

Portanto, com relação a essa coleta de dados, teve como desafio traçar caminhos para dois agentes de coleta, considerando uma divisão similar da quantidade de imóveis a serem coletados pelos agentes, e percorrerem uma distância iguais.

Com isso, foram traçados três métodos para esse cálculo utilizando Grafos, a primeira sendo mensurada apenas pela distância percorrida pelos dois agentes coletores, a segunda sendo mensurada pela quantidade de imóveis a serem coletados, e a terceira sendo uma média da distância percorrida com a quantidade de imóveis coletados pelo caminho, trazendo assim três estudos e análises de qual método será mais eficiente diante a problemática.

2 Referencial

Antes de aprofundar nos conceitos téncicos se torna fundamental estabelecer um estudo dos principais conceitos que são utilizados na definição do modelo e resolução do problema.

2.1 Conceitos de Grafos

A base teórica da resuolução do problema proposto fundamenta-se na teoria de grafos. Um grafo é formalmente definido como uma estrutura matemática $G = (V, E)$ onde V representa um conjunto de vértices e E um conjunto de arestas conectando pares de vértices. Quando cada aresta possui um peso associado, representando custo, distância, tempo ou outra métrica relevante, temos um grafo ponderado, que é fundamental para modelar problemas de otimização de rotas.

Um conceito central para o problema apresentado é o circuito euleriano, definido como um caminho fechado que visita cada aresta exatamente uma vez, e o passeio fechado, que é um caminho fechado que pode percorrer arestas múltiplas vezes. A existência de um circuito euleriano depende das propriedades do grafo, particularmente do grau de vértice, que representa o número de arestas incidentes a um vértice. A relação entre graus de vértices e a existência de circuitos eulerianos constitui um dos resultados fundamentais da teoria de grafos.

2.2 Origens do Problema Euleriano

A origem dos problemas de roteamento euleriano remonta ao século XVIII, quando o matemático suíço Leonhard Euler resolveu o célebre desafio das pontes de Königsberg em 1736 Euler (1736). Euler demonstrou que não era viável cruzar todas as sete pontes da cidade uma única vez, estabelecendo, assim, as bases da teoria dos grafos e dos circuitos eulerianos.

Depois, em 1873, Carl Hierholzer criou o primeiro algoritmo eficaz para localizar circuitos eulerianos em grafos conectados que possuem todos os vértices com grau par Hierholzer (1873). Este algoritmo, chamado de algoritmo de Hierholzer, continua sendo a fundamentação para a solução de problemas eulerianos até hoje.

2.2.1 Conceitos de Partição

A partição de um grafo refere-se à separação do conjunto de vértices em subconjuntos distintos Schaeffer (2007), sendo crucial para a alocação de tarefas entre vários agentes, onde o balanceamento diz respeito à distribuição justa de carga entre componentes diversos, assegurando que nenhum agente fique excessivamente ocupado enquanto outros permanecem sem atividade

Uma propriedade crucial para garantir a viabilidade das soluções é a conectividade, que representa a propriedade de um grafo ou subgrafo de possuir caminhos entre todos os pares de vértices. Esta propriedade assegura que cada agente possa acessar todos os vértices atribuídos a ele, sendo essencial para a aplicabilidade prática das soluções geradas.

2.2.2 Partição Balanceada de Grafos

A divisão de grafos é um desafio fundamental em otimização combinatória, com utilizações em computação paralela Kernighan & Lin (1970), balanceamento de carga Hendrickson & Leland (1995) e repartição de tarefas Schaeffer (2007). O desafio da partição balanceada é considerado NP-n, mesmo em casos simples Garey & Johnson (1979), o que justifica a utilização de heurísticas e algoritmos aproximados na prática Fiduccia & Mattheyses (1982).

Considerando um grafo $G = (V, E)$ que possui pesos nas suas arestas e um vértice raiz $r \in V$, o desafio da partição balanceada consiste em dividir V em dois subconjuntos A e B que atendam a três condições essenciais. Inicialmente, o vértice raiz deve fazer parte de ambos os conjuntos, ou seja, $r \in A \cap B$, assegurando que os dois agentes comecem do mesmo ponto de partida. Em segundo lugar, os subgrafos induzidos por A e B precisam ser conexos, garantindo que todos os agentes tenham acesso a todos os vértices em sua partição. Finalmente, a discrepância entre os pesos totais dos subgrafos deve ser reduzida, assegurando um balanceamento apropriado da carga de trabalho entre os agentes.

2.2.3 Algoritmo Rooted Balanced Partition

O algoritmo rooted balanced partition utilizado é uma heurística gulosa que processa os vértices em ordem crescente de distância ao vértice raiz, atribuindo cada vértice ao conjunto que resulta em menor desbalanceamento, sempre garantindo a conectividade Hendrickson & Leland (1995). A função de decisão considera o incremento de peso que cada vértice adicionaria a cada conjunto, calculado como a soma dos pesos das arestas incidentes.

Algorithm 1: Algoritmo Rooted Balanced Partition

1 [1] Grafo $G = (V, E)$ com pesos nas arestas, vértice raiz $r \in V$ Partição (A, B) de V tal que $r \in A \cap B$, A e B são conexos, e balanceamento é minimizado $A \leftarrow \{r\}$, $B \leftarrow \{r\}$ Calcular distâncias de todos os vértices até r Ordenar vértices $V \setminus \{r\}$ por distância crescente até r cada vértice v na ordem $peso_A \leftarrow$ soma dos pesos das arestas entre v e vértices em A $peso_B \leftarrow$ soma dos pesos das arestas entre v e vértices em B $peso_total_A \leftarrow$ peso total do subgrafo induzido por $A \cup \{v\}$ $peso_total_B \leftarrow$ peso total do subgrafo induzido por $B \cup \{v\}$ $|peso_total_A - peso_total_B| < |peso_total_B - peso_total_A|$ $A \leftarrow A \cup \{v\}$ $B \leftarrow B \cup \{v\}$ (A, B)

2.3 Problema do Carteiro Chinês

O Problema do Carteiro Chinês (Chinese Postman Problem - CPP), introduzido por Kwan Mei-Ko em 1962 Kwan (1962), tem como objetivo determinar o percurso fechado mais curto que visita todas as arestas de um grafo no mínimo uma vez, voltando ao vértice de partida. Esse problema possui aplicações práticas em roteamento de veículos, coleta de resíduos e verificação de redes.

Formalmente, para um grafo não direcionado $G = (V, E)$ com pesos nas arestas $w : E \to \mathbb{R}^+$, o CPP procura determinar um percurso circular de custo mínimo. Ao contrário de um circuito euleriano (que passa por cada aresta exatamente uma vez), o passeio fechado do CPP pode precisar atravessar algumas arestas várias vezes Biggs et al. (1993).

O método tradicional para solucionar o CPP envolve: (1) localizar vértices com grau ímpar, (2) criar um grafo completo entre esses vértices com pesos definidos pelas menores distâncias, (3) determinar um emparelhamento mínimo, (4) replicar as arestas dos caminhos mínimos associados, e (5) descobrir um passeio fechado através do algoritmo de Hierholzer Hierholzer (1873); Edmonds & Johnson (1973). A etapa de matching mínimo é a que determina a complexidade, sendo solucionável em tempo $O(|V|^3)$ Edmonds (1965).

O algoritmo clássico para resolver o CPP em grafos não direcionados pode ser descrito em cinco etapas principais:

1. **Identificação de vértices ímpares**: Identificar todos os vértices com grau ímpar no grafo.
2. **Construção do grafo completo**: Criar um grafo completo K entre os vértices ímpares, onde o peso de cada aresta (u, v) é a distância mínima entre u e v no grafo original.

Algorithm 2: Algoritmo do Problema do Carteiro Chinês

1 [1] Grafo não direcionado $G = (V, E)$ com pesos nas arestas Passeio fechado que percorre todas as arestas pelo menos uma vez $V_{impar} \leftarrow \{v \in V : \text{grau}(v) \text{ é ímpar}\}$ $V_{impar} = \emptyset$ Hierholzer(G) Construir grafo completo K entre vértices em V_{impar} com pesos dados pelas distâncias mínimas em G $M \leftarrow$ Matching mínimo em K cada par $(u, v) \in M$ $P \leftarrow$ Caminho mínimo entre u e v em G Duplicar todas as arestas de P em G Hierholzer(G)

3. **Matching mínimo**: Encontrar um emparelhamento perfeito de peso mínimo (minimum weight perfect matching) no grafo K.
4. **Duplicação de arestas**: Para cada par (u, v) no matching, duplicar as arestas do caminho mínimo entre u e v no grafo original.
5. **Passeio fechado**: Após a duplicação, o grafo torna-se euleriano, e um passeio euleriano pode ser encontrado usando algoritmos como o de Hierholzer. Este passeio corresponde à solução do CPP.

Variações do algoritmo têm sido propostas para melhorar o desempenho em casos práticos. Algoritmos aproximados com garantias de qualidade têm sido desenvolvidos para instâncias muito grandes.

2.3.1 Problema do Carteiro Chinês com Múltiplos Agentes

A extensão do Problema do Carteiro Chinês para múltiplos agentes tem recebido atenção crescente. Dror (2000) Dror (2000) estuda variantes do CPP com múltiplos veículos, apresentando heurísticas baseadas em partição do grafo. Pearn e Assad (1995) Pearn & Assad (1995) investigam o problema de roteamento de múltiplos veículos para inspeção de redes, combinando técnicas de partição com algoritmos eulerianos.

Trabalhos mais recentes exploram o CPP multi-agente com restrições adicionais. Monnot et al. (2003) Monnot et al. (2003) propõem algoritmos aproximados para o CPP com múltiplos agentes em grafos grandes.

2.4 Trabalhos Correlatos

O problema de dividir rotas entre múltiplos agentes tem sido extensivamente estudado na literatura de roteamento de veículos Laporte (2009). A extensão do Problema do Carteiro Chinês para múltiplos agentes tem recebido atenção crescente, com trabalhos como Dror (2000) Dror (2000) e Pearn e Assad (1995) Pearn & Assad (1995) investigando heurísticas baseadas em partição do grafo.

A aplicação de técnicas de partição balanceada em problemas de roteamento tem sido investigada em diversos contextos Hendrickson & Leland (1995). A combinação de múltiplos critérios em problemas

3 Desenvolvimento

Para a resolução do problema em questão, utilizamos uma modelagem com grafo simples, utilizando as interseções das ruas como vértices e as próprias ruas como arestas, as imagens a seguir mostram a área utilizada como referência para modelagem e como ficou a modelagem do grafo:

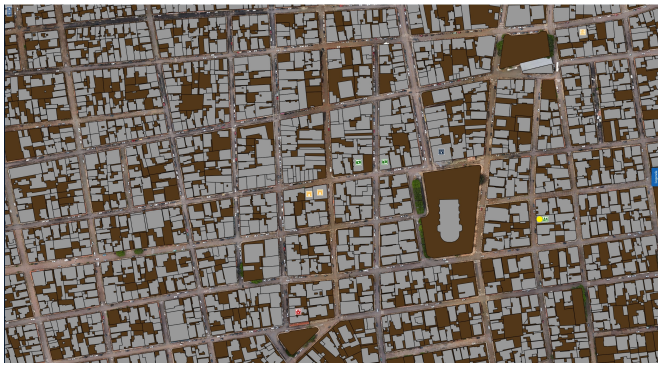

Figura 1: Área utilizada como modelagem, região de Elói Mendes/MG.

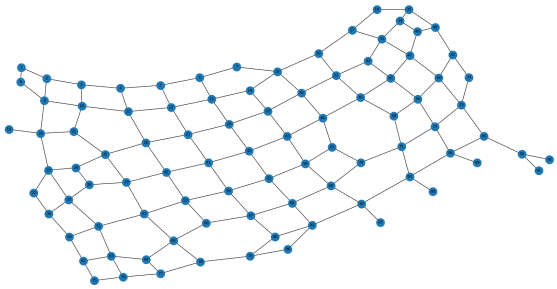

Figura 2: Grafo da Área utilizada como modelo.

3.1 Modelagem (características do dataset)

Após feita a modelagem inicial e construção do grafo, constatamos que haviam 92 vértices 151 arestas presentes no grafo e precisariamos agora estabelecer qual o critério de pesos adotariamos para a resolução do problema.

Para os pesos adotamos a distancia das ruas e a quantidade de casas, utlizando o site `https://geojson.io/#map=15.58/-21.610066/-45.567381/-10.4` para a medição das ruas e as próprias imagens fornecidas pelo professor para a quantidade de casas.

Ambas as informações foram utilizadas como pesos, pois influenciam diretamente o custo associado ao deslocamento ou à coleta em cada rua. A quantidade de casas funciona como um proxy para o *esforço operacional*, enquanto a distância representa o *esforço físico* de deslocamento.

Após a escolha dos pesos, conseguimos obter 3 modelos para efeito de comparação da resolução do problema:

a. Modelo A, utilizando os 2 pesos combinados.

b. Modelo B, utilizando apenas a distancia da rua como peso.

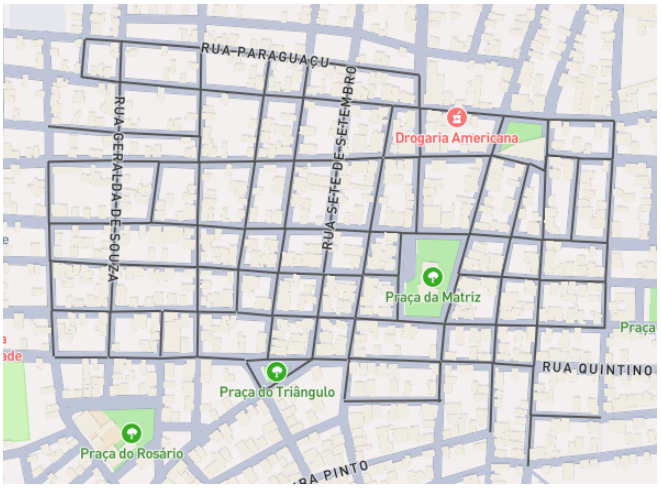

Figura 3: Mapa da Area de Elói Mendes no site geojson.io As linhas em cinza no mapa indicam as medições feitas das ruas.

c. Modelo C, utilizando apenas a quantidade de casas como peso.

Para a construção do grafo do modelo A, foi necessária realizar a normalização dos pesos pois distância e número de casas estão em escalas totalmente diferentes (por exemplo, 4 casas versus 300 metros). Sem normalização, um dos atributos dominaria o cálculo.

O uso do método de **Normalização Min-Max** torna possível equilibrar atributos heterogêneos, controlar a importância relativa de cada fator, adaptar o grafo a diferentes cenários operacionais. Assim, o grafo final é ponderado por um custo híbrido mais realista.

Como cada aresta possui dois atributos distintos, foi necessário construir um peso combinado. Para isso, definimos dois coeficientes:

- **(alpha):** peso associado à distância
- **(beta):** peso associado à quantidade de casas

O peso final de cada aresta é calculado como:

$$w(u,v) = \alpha \cdot \text{dist}(u,v) + \beta \cdot \text{casas}(u,v) \tag{1}$$

No projeto foi adotado:

- **alpha = 0.6**
- **beta = 0.4**

Esses valores foram escolhidos com base na interpretação prática do problema: a distância tem impacto ligeiramente maior no tempo de execução das tarefas, mas ainda é importante penalizar vias com muitas casas devido à carga de trabalho operacional envolvida.

Para melhor compreensão do impacto dos pesos nos modelos aplicamos a função de **Mapa de cores 'Inferno' (Matplotlib colormap)** , onde as arestas com cores mais claras e largas indicam peso maior e as mais escuras e estreitas indicam peso menor.

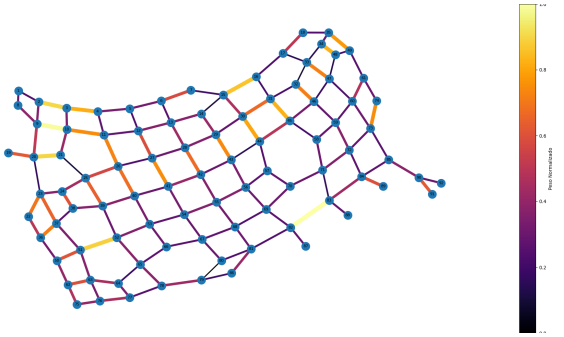

Figura 4: Grafo Modelo A

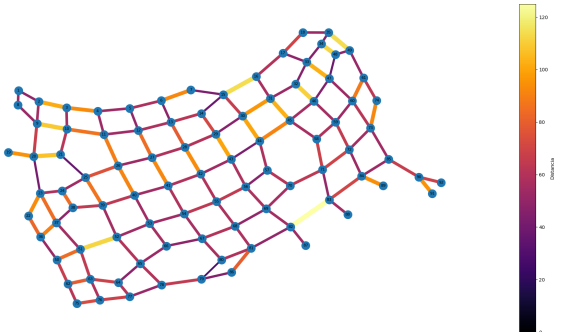

Figura 5: Grafo Modelo B

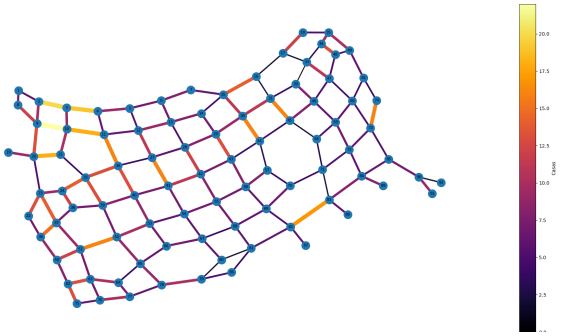

Figura 6: Grafo Modelo C

3.2 Método de Solução (Detalhamento)

Tendo os grafos dos 3 modelos prontos, vamos agora detalhar a forma de implementação para a divisão dos grafos para 2 agentes (2 subgrafos), utilização do algoritmo CCP (Caminho de Cobertura Prioritária) para definir o caminho mais eficiente para cada agente e o tratamento entre os clusters que ligam os 2 subgrafos.

Para a divisão do grafo em dois subconjuntos aproximadamente equilibrados, foi utilizado um método inspirado em técnicas de **Balanced Graph Partitioning**, adotando uma abordagem **Rooted** (enraizada), isto é, partindo de um vértice inicial fixo que define uma ordem de expansão do cluster.

Como o grafo representa uma malha urbana com distribuição espacial irregular, um método simples como *k-means* não é adequado. Assim, adotou-se uma estratégia baseada na expansão controlada de uma BFS.

Código 1: Implementação do algoritmo Rooted Balanced Partition

```
def rooted_balanced_partition(G, root, alpha=0.6, beta=0.4):

    # Normalizar distancias e casas
    dn = normalize([G[u][v]["distancia"] for u, v in G.edges()])
    cn = normalize([G[u][v]["casas"] for u, v in G.edges()])

    for (edge, dnorm, cnorm) in zip(G.edges(), dn, cn):
        u, v = edge
        G[u][v]["peso"] = alpha*dnorm + beta*cnorm

    # distancias ao root
    dist_root = nx.single_source_dijkstra_path_length(G, root, weight="peso")
    ordem = sorted(dist_root, key=lambda x: dist_root[x])

    A = set([root])
    B = set([root])
    pesoA = 0
    pesoB = 0

    def can_add(S, v):
        S2 = S | {v}
        return nx.has_path(G.subgraph(S2), root, v)

    for v in ordem:
        if v == root:
            continue

        incA = sum(G[v][u]["peso"] for u in G.neighbors(v) if u in A)
        incB = sum(G[v][u]["peso"] for u in G.neighbors(v) if u in B)

        if pesoA <= pesoB:
            if can_add(A, v):
                A.add(v)
                pesoA += incA
            else:
                B.add(v)
                pesoB += incB
        else:
            if can_add(B, v):
                B.add(v)
                pesoB += incB
            else:
                A.add(v)
                pesoA += incA

    return A, B

root = 71
A, B = rooted_balanced_partition(G, root=root)
```

Depois da divisão obtida, surgem arestas que ligam os subconjuntos A e B. Essas arestas são chamadas de **arestas de fronteira** (ou *cut edges*). É necessário tratar estas arestas, para que não atrapalhe na atribuição de rotas aos agentes.

Primeiro identificamos as arestas de cortes com a seguinte operação:

$$E_{\text{corte}} = \{\, (u,v) \in E \mid u \in A,\ v \in B \,\}$$

Depois Realizamos o Tratamento das arestas com duas opções: **Replicação do custo na fronteira ou Reforço das conexões**(caso o cluster vire não-euleriano)**.** Na Resplicação do custo da fronteira cada agente mantêm apenas sua parte, e apenas as arestas internas são usadas para geração da rota euleriana. Já no Reforço das conexões, se um dos clusters estiver com grau ímpar demais, é feita a duplicação de algumas arestas internas ou reconexão mínima (com menor peso) para torná-lo euleriano. Esse tratamento é crucial para permitir que cada cluster execute o **algoritmo do Carteiro Chinês (CPP)** separadamente.

Código 2: Implementação do tratamento entre clusters

```
edges_AB = [(u, v) for u, v in G.edges()
            if (u in A_vertices and v in
                B_vertices) or
               (u in B_vertices and v in
                   A_vertices)]

for u, v in edges_AB:
    # Distancias do START ate as
        extremidades
    d_u = nx.shortest_path_length(G,
        START, u, weight="peso")
    d_v = nx.shortest_path_length(G,
        START, v, weight="peso")

    # Aresta vai para o cluster mais
        pr ximo do START
    if d_u < d_v:
        if u in A_vertices:
            G_A.add_edge(u, v, **G[u][v])
        else:
            G_B.add_edge(u, v, **G[u][v])
    else:
        if v in A_vertices:
            G_A.add_edge(u, v, **G[u][v])
        else:
            G_B.add_edge(u, v, **G[u][v])
```

O **algoritmo CCP (Caminho de Cobertura Prioritária)** tem como objetivo determinar uma rota que percorra o grafo de forma eficiente, priorizando arestas de maior importância conforme um critério definido pelo problema. No contexto do projeto, essa importância é dada pelos **pesos das arestas**, calculados a partir da combinação:

$$w(u,v) = \alpha \cdot dist(u,v) + \beta \cdot casas \tag{2}$$

Onde:

- **dist(u,v)** = distância física da rua (em metros);
- **casas(u,v)** = quantidade de casas naquela rua;
- **alpha e beta** = parâmetros de normalização que balanceiam a influência das duas variáveis.

Código 3: Implementação do algoritmo CCP

```
def chinese_postman(G_sub, start):
    # Garantir que start esteja no
        subgrafo
    if start not in G_sub:
        start = list(G_sub.nodes())[0]  #
            usa outro n

    # 1. Vertices impares
    odd = [v for v in G_sub.nodes() if
        G_sub.degree(v) % 2 == 1]

    # 2. Grafo completo entre impares (
        distancias reais)
    K = nx.Graph()
    for u, v in itertools.combinations(
        odd, 2):
        dist = nx.shortest_path_length(
            G_sub, u, v, weight="peso")
        K.add_edge(u, v, weight=dist)

    # 3. Matching minimo
    M = nx.min_weight_matching(K, weight=
        "weight")

    # 4. Duplicacao correta das arestas
        do caminho minimo
    G_euler = nx.MultiGraph(G_sub)

    for u, v in M:
        path = nx.shortest_path(G_sub, u,
            v, weight="peso")
        for a, b in zip(path, path[1:]):
            G_euler.add_edge(a, b, peso=
                G_sub[a][b]["peso"])

    # 5. Agora Euleriano
    circuito = list(nx.eulerian_circuit(
        G_euler, source=start))

    rota = [start]
    for u, v in circuito:
        rota.append(v)

    return rota
```

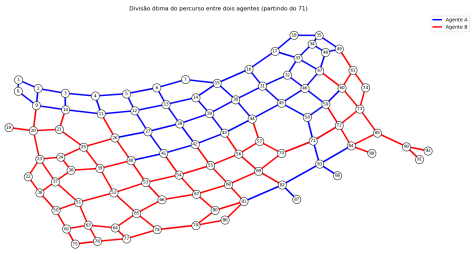

Figura 7: Grafo Modelo A dividido com melhor caminho

4 Resultados

4.1 Métricas de Avaliação

As principais métricas utilizadas para avaliar as soluções foram selecionadas para capturar diferentes aspectos da eficiência das abordagens propostas, onde o tempo de um agente representa o tempo necessário para um único agente completar

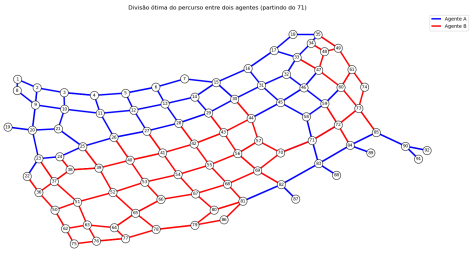

Figura 8: Grafo Modelo B dividido com melhor caminho

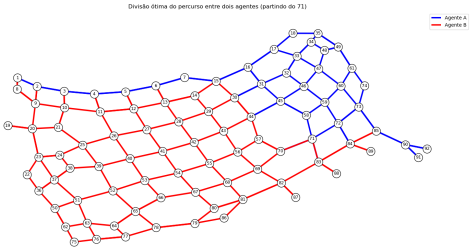

Figura 9: Grafo Modelo C dividido com melhor caminho

todo o trabalho, servindo como linha de base para comparação. Os tempos do Agente A e do Agente B indicam o tempo necessário para cada agente completar sua rota específica, permitindo avaliar o balanceamento da carga de trabalho.

O tempo da equipe, calculado como o máximo entre os tempos dos dois agentes, representa o makespan do sistema e indica o tempo total necessário para completar todo o trabalho quando dois agentes trabalham em paralelo. A economia de tempo é obtida pela diferença entre o tempo de um agente e o tempo da equipe, quantificando o ganho absoluto obtido com a divisão de trabalho. Por fim, a redução percentual expressa este ganho em termos relativos, mostrando o percentual de redução do tempo ao utilizar dois agentes em vez de um único.

4.2 Resultados por Cenário

4.2.1 Cenário 1: Avaliação por Casas

No cenário de avaliação por casas, os resultados demonstram a eficácia da divisão de trabalho quando o critério principal é o número de casas a serem visitadas. A partição balanceada garante que cada agente receba aproximadamente a mesma carga de trabalho em termos de número de casas, promovendo uma distribuição equitativa das responsabilidades.

A distribuição de casas entre os agentes apresenta-se aproximadamente equilibrada, indicando que o algoritmo de partição consegue efetivamente balancear a carga de trabalho mesmo quando apenas um critério é considerado. As rotas geradas pelo algoritmo CPP garantem cobertura completa de todas as arestas, assegurando que nenhuma localização seja negligenciada.

A Tabela 1 apresenta os resultados comparativos entre o desempenho de um único agente e a solução utilizando dois agentes para este cenário. Esta comparação permite avaliar o ganho de eficiência obtido através da divisão de trabalho quando o critério principal é o número de casas a serem visitadas.

Métrica	1 Agente	2 Agentes
Tempo total (min)	643.00	444.50
Tempo Agente A (min)	–	444.50
Tempo Agente B (min)	–	155.50
Economia de tempo (min)	–	198.50
Redução percentual (%)	–	30.87%

Tabela 1: Comparação 1 Agente vs 2 Agentes - Avaliação por Casas

Os resultados apresentados na Tabela 1 revelam que a utilização de dois agentes resulta em uma redução de tempo de 30.87%, correspondendo a uma economia de 198.50 minutos. Observa-se, no entanto, um desbalanceamento significativo entre os agentes, com o Agente A executando uma carga de trabalho substancialmente maior (444.50 minutos) em comparação ao Agente B (155.50 minutos). Este desbalanceamento indica que, embora a partição por casas promova uma distribuição equitativa do número de casas, a modelagem de tempo que considera tanto o número de casas quanto a distância percorrida resulta em diferenças significativas no tempo total de trabalho de cada agente.

4.2.2 Cenário 2: Avaliação por Distância

No cenário de avaliação por distância, o foco está em minimizar o deslocamento total. Este cenário é particularmente relevante quando o tempo de deslocamento é o fator dominante, como em áreas rurais ou com baixa densidade de casas, onde o tempo gasto em trânsito supera significativamente o tempo de registro do imóvel.

A partição neste cenário considera principalmente a proximidade geográfica dos vértices, agrupando vértices próximos no mesmo subgrafo para minimizar distâncias de deslocamento. As rotas são otimizadas para minimizar distâncias percorridas, resultando em trajetos mais compactos e eficientes.

A Tabela 2 apresenta os resultados comparativos entre o desempenho de um único agente e a solução utilizando dois agentes para este cenário. Esta análise é particularmente relevante para avaliar a eficiência da divisão de trabalho quando o foco está em minimizar distâncias percorridas.

Métrica	1 Agente	2 Agentes
Tempo total (min)	74.45	32.87
Tempo Agente A (min)	–	32.87
Tempo Agente B (min)	–	29.46
Economia de tempo (min)	–	41.58
Redução percentual (%)	–	55.85%

Tabela 2: Comparação 1 Agente vs 2 Agentes - Avaliação por Distância

Os resultados apresentados na Tabela 2 demonstram que o cenário de avaliação por distância apresenta a maior redução percentual de tempo entre os três cenários analisados, alcançando 55.85% de redução. Esta redução corresponde a uma economia de 41.58 minutos. Além disso, observa-se um exce-

lente balanceamento entre os agentes, com o Agente A executando 32.87 minutos e o Agente B executando 29.46 minutos, resultando em uma diferença de apenas 3.41 minutos. Este balanceamento superior indica que a partição baseada em distância promove uma distribuição mais equilibrada do tempo de trabalho quando comparada à partição baseada apenas no número de casas.

4.2.3 Cenário 3: Avaliação Combinada

O cenário combinado representa a situação mais realista, onde tanto a distância quanto o número de casas são fatores importantes. A combinação ponderada (60% distância, 40% casas) permite uma avaliação mais equilibrada que considera múltiplos aspectos simultaneamente.

A normalização é essencial para combinar métricas de diferentes escalas, transformando distâncias medidas em metros e números de casas em valores comparáveis no intervalo $[0, 1]$. Os pesos $\alpha = 0.6$ e $\beta = 0.4$ refletem a importância relativa de cada critério, atribuindo maior peso à distância enquanto ainda considera significativamente o número de casas.

A Tabela 3 apresenta os resultados comparativos entre o desempenho de um único agente e a solução utilizando dois agentes para este cenário. Esta tabela permite avaliar o desempenho da abordagem mais realista, que combina múltiplos critérios de avaliação.

Métrica	1 Agente	2 Agentes
Tempo total (min)	684.45	321.11
Tempo Agente A (min)	–	297.18
Tempo Agente B (min)	–	321.11
Economia de tempo (min)	–	363.35
Redução percentual (%)	–	53.09%

Tabela 3: Comparação 1 Agente vs 2 Agentes - Avaliação Combinada

Os resultados apresentados na Tabela 3 demonstram que o cenário combinado oferece uma redução de tempo de 53.09%, correspondendo à maior economia absoluta de tempo entre os três cenários (363.35 minutos). O balanceamento entre os agentes é satisfatório, com o Agente A executando 297.18 minutos e o Agente B executando 321.11 minutos, resultando em uma diferença de 23.93 minutos. Este cenário representa um compromisso entre eficiência de deslocamento e distribuição de carga, resultando em soluções que são simultaneamente eficientes em termos de tempo de viagem e razoavelmente balanceadas em termos de carga de trabalho.

4.3 Análise Comparativa dos Cenários

A Tabela 4 apresenta uma comparação transversal dos três cenários, permitindo avaliar como diferentes critérios de avaliação influenciam os resultados. Esta análise é fundamental para identificar qual abordagem oferece o melhor desempenho em diferentes métricas e para entender os trade-offs entre diferentes critérios de avaliação.

A análise comparativa apresentada na Tabela 4 permite identificar padrões importantes nos resultados. Primeiramente, observa-se que a utilização de dois agentes resulta consistentemente em redução significativa do tempo total de trabalho em todos os cenários, validando a eficácia da abordagem proposta.

Métrica	Casas	Distância	Combinada
Tempo 1 agente (min)	643.00	74.45	684.45
Tempo Agente A (min)	444.50	32.87	297.18
Tempo Agente B (min)	155.50	29.46	321.11
Makespan - 2 agentes (min)	444.50	32.87	321.11
Economia de tempo (min)	198.50	41.58	363.35
Redução percentual (%)	30.87%	55.85%	53.09%

Tabela 4: Comparação da Avaliação dos Pesos entre os Três Cenários

Em relação à redução percentual, o cenário de avaliação por distância apresenta o melhor desempenho (55.85%), seguido pelo cenário combinado (53.09%) e pelo cenário de avaliação por casas (30.87%). Esta diferença pode ser atribuída ao fato de que a partição baseada em distância promove um melhor balanceamento do tempo de trabalho entre os agentes, enquanto a partição baseada apenas em casas resulta em desbalanceamento significativo.

Quanto à economia absoluta de tempo, o cenário combinado apresenta a maior economia (363.35 minutos), seguido pelo cenário de avaliação por casas (198.50 minutos) e pelo cenário de avaliação por distância (41.58 minutos). Esta diferença reflete o tempo total de trabalho de um único agente em cada cenário, que varia significativamente conforme o critério de avaliação utilizado.

Em relação ao balanceamento entre agentes, o cenário de avaliação por distância apresenta o melhor desempenho, com uma diferença de apenas 3.41 minutos entre os agentes. O cenário combinado apresenta balanceamento satisfatório (diferença de 23.93 minutos), enquanto o cenário de avaliação por casas apresenta desbalanceamento significativo (diferença de 289.00 minutos).

4.4 Discussão dos Resultados

Os resultados demonstram que a abordagem proposta é eficaz para otimizar a divisão de trabalho entre múltiplos agentes. A eficiência da partição é evidenciada pela capacidade do algoritmo rooted balanced partition em dividir o grafo de forma equilibrada, garantindo que ambos os agentes tenham cargas de trabalho similares quando critérios adequados são utilizados. Esta capacidade de balanceamento é fundamental para maximizar a utilização dos recursos disponíveis.

A otimização das rotas é alcançada através do algoritmo CPP, que garante que cada agente percorra todas as arestas de seu subgrafo de forma eficiente, minimizando o custo total. Esta otimização é particularmente importante em contextos onde o custo de deslocamento é significativo.

A redução de tempo obtida através da utilização de dois agentes resulta em redução significativa do tempo total de trabalho, variando de 30.87% a 55.85% conforme o critério de avaliação utilizado. Este resultado valida a hipótese de que a paralelização do trabalho pode gerar ganhos substanciais de eficiência, especialmente quando critérios de avaliação adequados são selecionados.

A influência dos critérios de avaliação é claramente observada nos resultados, onde diferentes critérios resultam em partições distintas, cada uma otimizada para seu objetivo específico. Esta sensibilidade aos critérios demonstra a impor-

tância de selecionar adequadamente as métricas de avaliação conforme o contexto da aplicação e os objetivos prioritários (balanceamento, economia de tempo, ou compromisso entre ambos).

5 Conclusões

O presente projeto teve como objetivo modelar a malha urbana de um bairro como um grafo ponderado e propor uma metodologia eficiente para divisão equilibrada do território entre dois agentes responsáveis pela coleta de dados imobiliários. A partir de um conjunto composto por 92 vértices (interseções) e 151 arestas (ruas), foi possível estruturar formalmente o problema, atribuindo pesos às arestas com base em duas variáveis essenciais para o esforço operacional: a distância física e a quantidade de casas presentes em cada rua.

A normalização dos atributos e a construção do peso composto permitiram harmonizar grandezas distintas e ajustar a relevância relativa de cada fator por meio dos parâmetros Alpha e Beta. O uso dessa ponderação proporcionou uma representação mais realista do custo associado ao trabalho dos agentes, refletindo tanto o deslocamento necessário quanto o volume de atendimentos.

Para identificar regiões equilibradas, foi implementado o algoritmo Rooted Balanced Partition, que, a partir de um vértice raiz estrategicamente escolhido, distribuiu o grafo em dois subconjuntos conectados. Tal método garantiu que cada agente recebesse uma área contínua e operacionalmente viável. Além disso, foi analisado o conjunto de arestas de corte, essenciais para compreender as transições entre clusters e para avaliar a coerência da divisão.

A modelagem também envolveu a construção de histogramas, mapas visuais e grafos coloridos que auxiliaram na interpretação da distribuição dos pesos, destacando ruas com maiores concentrações de casas ou maiores distâncias. Esses elementos gráficos foram fundamentais para verificar a coerência da modelagem e identificar pontos críticos da rede.

Em síntese, o projeto cumpriu os objetivos iniciais ao entregar uma modelagem consistente, uma estratégia de particionamento conectada e balanceada, e uma análise quantitativa clara do esforço operacional. A combinação de técnicas de teoria dos grafos, normalização de atributos e algoritmos de particionamento evidenciou o potencial da modelagem matemática como ferramenta de apoio à tomada de decisão em contextos urbanos e logísticos.

Referências

Biggs, N. L., Lloyd, E. K., & Wilson, R. J. (1993). *Graph Theory 1736-1936*. Oxford University Press.

Dror, M. (2000). *Arc Routing: Theory, Solutions and Applications*. Kluwer Academic Publishers.

Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17, 449–467.

Edmonds, J. & Johnson, E. L. (1973). Matching, euler tours and the chinese postman. *Mathematical Programming*, 5(1), 88–124.

Euler, L. (1736). Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Petropolitanae*, 8, 128–140.

Fiduccia, C. M. & Mattheyses, R. M. (1982). A linear-time heuristic for improving network partitions. Em *Proceedings of the 19th Design Automation Conference* (pp. 175–181).

Garey, M. R. & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Hendrickson, B. & Leland, R. (1995). A multilevel algorithm for partitioning graphs. Em *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing* (pp.̃28).

Hierholzer, C. (1873). Über die möglichkeit, einen linienzug ohne wiederholung und ohne unterbrechung zu umfahren. *Mathematische Annalen*, 6(1), 30–32.

Kernighan, B. W. & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2), 291–307.

Kwan, M.-K. (1962). Graphic programming using odd or even points. *Chinese Mathematics*, 1, 273–277.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4), 408–416.

Monnot, J., Toulouse, S., & Wei, B. Y. (2003). Approximation results for the minimum k-chinese postman problem. *Operations Research Letters*, 31(5), 357–365.

Pearn, W. L. & Assad, A. A. (1995). Routing vehicles in the chinese postman problem. *Operations Research*, 43(5), 843–851.

Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, 1(1), 27–64.