CENTRO UNIVERSITÁRIO IESB

ENGENHARIA DE SOFTWARE

CARLOS EDUARDO DE SOUZA LEMOS

2572190056

Teoria dos Grafos e Análise de Algoritmos

Brasília - DF

Setembro – 2025

Relatório Técnico – Teoria dos Grafos e Análise de Algoritmos

1. Introdução

O que é um Grafo

Um **grafo** é uma estrutura matemática utilizada para representar relações entre objetos. Formalmente, um grafo é definido como um par $G = (V, A)$, em que **V** representa o conjunto de vértices (ou nós) e **A** representa o conjunto de arestas (ou arcos), que conectam pares de vértices. Os grafos podem assumir diferentes formas, como os **direcionados**, em que as arestas possuem uma direção específica; os **não-direcionados**, em que a conexão entre vértices é bidirecional; os **ponderados**, que atribuem pesos às arestas; e os **simples**, que não possuem laços ou arestas paralelas. Além disso, existem grafos **bipartidos**, **completos**, **árvores**, entre outros tipos, cada um com características próprias e aplicações específicas.

Características Principais

Entre as principais propriedades dos grafos, destaca-se a **adjacência**, que ocorre quando dois vértices estão conectados por uma aresta. O **grau de um vértice** corresponde ao número de arestas que incidem sobre ele. Os grafos também podem conter **caminhos** e **ciclos**, que são sequências de vértices e arestas que conectam diferentes pontos dentro da estrutura. A **conectividade** é outra característica importante, indicando se há caminhos entre todos os pares de vértices, tornando o grafo conexo. Quanto à representação, os grafos podem ser descritos por meio de **matrizes de adjacência** ou **listas de adjacência**, dependendo da aplicação e da eficiência desejada.

Aplicações de Grafos

Os grafos são amplamente utilizados em diversas áreas do conhecimento. Na **ciência da computação**, são fundamentais para o desenvolvimento de algoritmos de busca, como DFS (Depth-First Search) e BFS (Breadth-First Search), além de algoritmos de caminhos mínimos e árvores geradoras, como Dijkstra, Kruskal e Prim. Em **redes de computadores**, os grafos modelam conexões e roteamentos entre dispositivos. Na **engenharia e transporte**, são usados para otimizar rotas e fluxos logísticos. Na **biologia**, ajudam a representar interações entre espécies ou estruturas moleculares. Em **redes sociais**, permitem a análise de conexões entre usuários, enquanto em **Big Data**, são úteis para estruturar e visualizar dados complexos de forma eficiente.

2. Arquitetura Proposta

Este trabalho apresenta um software desenvolvido para a modelagem e análise de rotas de entrega com base em grafos. O sistema representa cada ponto de entrega como um vértice e as conexões entre eles como arestas ponderadas, possibilitando a visualização da rede, o cálculo de caminhos mínimos entre pares de nós e a obtenção de uma rota aproximada para o problema do caixeiro viajante (TSP). Além disso, a ferramenta é capaz de converter os custos das rotas em valores monetários de referência, reforçando a aplicação prática da solução.

A implementação foi realizada em Python 3.x, utilizando principalmente a biblioteca NetworkX, responsável pelo suporte à modelagem de grafos e pela execução de algoritmos específicos. Entre os recursos empregados estão a criação de nós e arestas ponderadas, a determinação de posições para visualização por meio do layout de forças (spring layout), a extração de pesos das arestas e o uso do algoritmo de Dijkstra para calcular caminhos mínimos e custos associados. Também foi utilizada a heurística disponibilizada pela própria biblioteca para tratar o problema do caixeiro viajante. Para a visualização dos resultados, a biblioteca Matplotlib foi incorporada, permitindo a geração de gráficos representativos do grafo com seus vértices, arestas e rótulos de pesos. Opcionalmente, o sistema pode ser integrado ao python-docx, possibilitando a exportação dos resultados em relatórios no formato .docx.

Na lógica do sistema, cada local de entrega é representado por um nó do grafo e cada conexão entre locais é descrita por uma tupla que indica origem, destino e peso, sendo este último associado ao custo ou à distância da ligação. A visualização é obtida a partir do cálculo automático de posições dos nós, que são desenhados junto com as arestas e seus respectivos pesos, o que facilita a análise topológica da rede de entregas.

Entre as funcionalidades implementadas, destacam-se a visualização do grafo, o cálculo de caminhos mínimos entre dois pontos ou de um ponto para todos os demais através do algoritmo de Dijkstra, a aproximação de uma solução para o problema do caixeiro viajante por meio da heurística do NetworkX e a conversão de custos de rotas em valores monetários a partir de um multiplicador fixo. O fluxo de execução do software inicia-se com a definição dos pontos de entrega e das conexões entre eles, prossegue com a construção do grafo e sua plotagem, executa o cálculo dos caminhos mínimos e da rota aproximada do TSP e, por fim, apresenta os resultados. De forma opcional, esses dados podem ser exportados para relatórios formatados e o multiplicador monetário pode ser ajustado conforme a necessidade.

Os resultados obtidos permitem identificar rotas de menor custo entre pares de pontos, úteis para o planejamento de entregas ponto a ponto, e também fornecem

uma solução aproximada para o problema do caixeiro viajante, que, embora não garanta otimalidade global por se tratar de uma heurística, é adequada para instâncias de tamanho médio e pode servir como base para aprimoramentos. A representação visual do grafo também se mostra importante, pois facilita a verificação manual das conexões e a comunicação dos resultados a diferentes públicos.

Apesar dos benefícios, o software apresenta algumas limitações, como a dependência de uma heurística para o TSP, que pode ser substituída por algoritmos exatos em instâncias menores, e o uso de grafos não direcionados, o que pode limitar a análise em casos em que os custos de ida e volta não são equivalentes. Entre as melhorias futuras, destacam-se a integração com dados reais de geolocalização, o uso de APIs de mapas para distâncias reais, a consideração de restrições adicionais como janelas temporais e capacidade de veículos, além do desenvolvimento de uma interface mais amigável, seja em linha de comando ou gráfica.

Para a utilização do software, é necessário instalar previamente as bibliotecas NetworkX e Matplotlib, além do python-docx caso haja interesse na exportação de relatórios. Em seguida, basta executar o script principal, que exibirá o grafo e imprimirá os resultados no terminal, sendo possível também personalizar a lista de nós e arestas conforme o caso de estudo.

---

3. Resultados

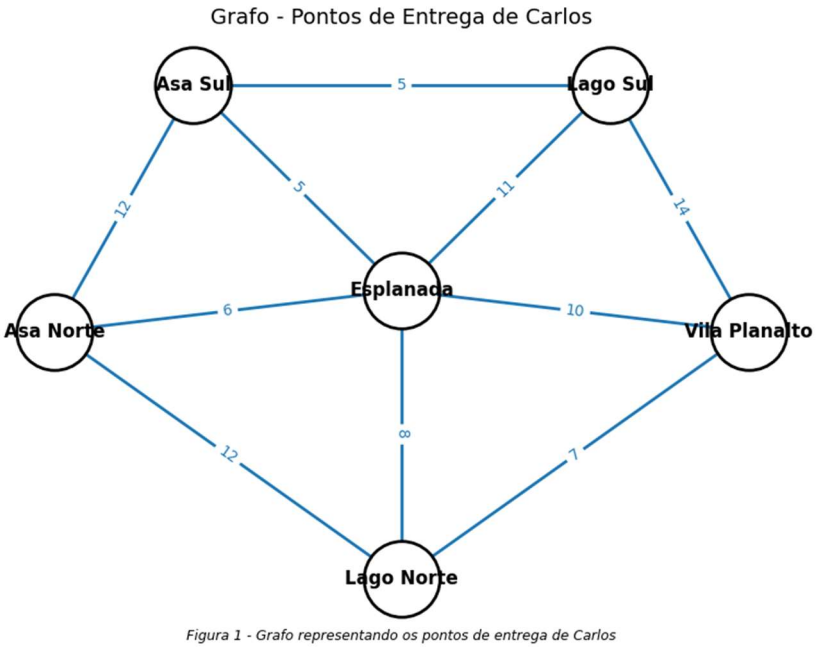

Figura 1 - Grafo representando os pontos de entrega de Carlos

Figura 1

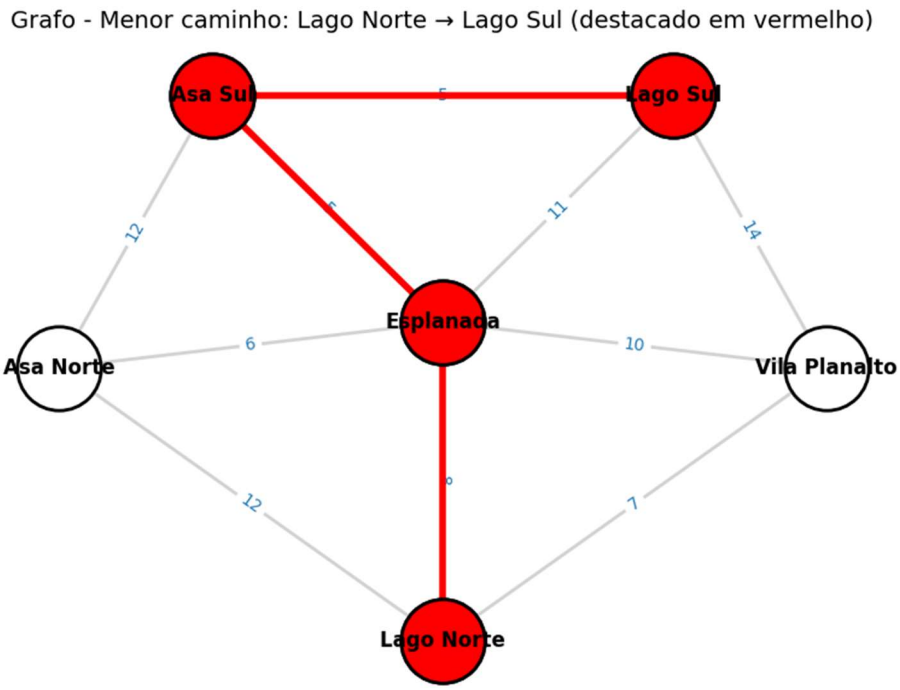

Figura 2

A metodologia adotada consistiu no uso dos menores caminhos entre os nós, obtidos por meio do algoritmo de Dijkstra, considerando tanto as distâncias quanto os custos associados. O problema de visitar todos os pontos foi tratado como uma instância do Problema do Caixeiro Viajante (TSP) e resolvido por meio das rotinas de aproximação do NetworkX (fechamento métrico / heurísticas disponibles), que geram uma rota aproximada e seu custo total. Esta abordagem é adequada para a instância em estudo e permite obter soluções rápidas; para garantia de otimalidade em casos maiores, recomenda-se o uso de algoritmos exatos ou solvers de otimização.

Inicialmente, calculou-se o custo dos menores caminhos partindo da sede, localizada no Lago Norte, até cada ponto de distribuição usando Dijkstra. Os menores custos obtidos a partir do Lago Norte, com as arestas atualmente definidas, foram:

- Lago Norte → Lago Norte: 0
- Lago Norte → Vila Planalto: 7
- Lago Norte → Esplanada: 8
- Lago Norte → Asa Norte: 12
- Lago Norte → Asa Sul: 13
- Lago Norte → Lago Sul: 18

A soma desses valores — que representa o custo agregado das viagens individuais do Lago Norte a cada destino, feitas separadamente — é 58 unidades de custo.

Ao considerar a necessidade de percorrer todos os pontos em uma única rota que sai e retorna ao Lago Norte, o script utiliza a função de TSP do NetworkX para obter uma solução aproximada (ciclo). O programa imprime a sequência de nós da rota aproximada e o custo total correspondente; como a solução é gerada por heurística sobre o fechamento métrico, a rota exibida e o custo podem variar conforme a heurística escolhida/implementação, mas são soluções práticas para planejamento em instâncias de dimensão moderada.

Para o cálculo monetário, o próprio script multiplica o custo do ciclo (ou de qualquer rota gerada) pelo valor unitário informado (no código, R$ 20,00 por unidade). Assim, para a rota que inicia na Esplanada e percorre todos os pontos até retornar ao Lago Norte, o valor em reais é obtido automaticamente pelo programa (variável tsp_esplanada_custo × R$ 20,00). Com os pesos atuais do grafo, a rota encontrada que parte da Esplanada e visita todos os pontos até retornar à sede no Lago Norte é: Esplanada → Asa Norte → Asa Sul → Lago Sul → Vila Planalto → Lago Norte; custo total = 43 unidades. Considerando R$ 20,00 por unidade, o custo monetário desse percurso é 43 × R$ 20,00 = R$ 860,00.

Por fim, ressalta-se que os cálculos assumem custos aditivos e simétricos (grafo não direcionado). A abordagem empregada é adequada para análise exploratória e prototipagem; para aplicações reais com maiores instâncias ou restrições operacionais (janelas de entrega, capacidades, assimetrias) recomenda-se adotar modelagem e solucionadores específicos.

---

4. Conclusão

O desenvolvimento deste trabalho permitiu compreender de forma prática a aplicação da teoria dos grafos em problemas reais de transporte e logística. A utilização da linguagem Python e das bibliotecas NetworkX e Matplotlib proporcionou uma abordagem eficiente tanto na modelagem quanto na visualização do grafo.

Entre as principais dificuldades enfrentadas, destacam-se a implementação do Problema do Caixeiro Viajante e o tratamento de caminhos aproximados, já que nem sempre há rotas diretas entre todos os vértices, e a familiarização com as bibliotecas utilizadas, compreendendo como funcionam e como aplicá-las corretamente na implementação do código. No entanto, a atividade contribuiu significativamente para o aprendizado sobre algoritmos de caminhos mínimos, otimização de rotas e análise de custos em redes.

Conforme solicitado, o código-fonte desenvolvido também será entregue juntamente com este relatório. Além disso, encontra-se disponível no repositório GitHub pelo link: https://github.com/CarlosEduardoLemos/Grafos

Bibliografia

1. TILAWAT, Midhat. *O que é Grafo (Matemática Discreta)?*. All About AI, 2024. Disponível em: https://www.allaboutai.com/pt-br/glossario-inteligencia-artificial/grafo-matematica-discreta. Acesso em: 08 set. 2025.[1]
2. IME-USP. *O que é um grafo?*. Instituto de Matemática e Estatística da USP. Disponível em: https://www.ime.usp.br/~pf/algoritmos*em*grafos/aulas/grafos.html. Acesso em: 08 set. 2025.
3. NETTO, Guilherme Tomaschewski. *Grafos: representação e aplicações*. Universidade Federal de Pelotas. Disponível em: https://netto.ufpel.edu.br/lib/exe/fetch.php?media=grafos.pdf. Acesso em: 08 set. 2025.
4. ESTATÍSTICA FÁCIL. *O que é: Grafos – Entenda a Estrutura e Aplicações*. Disponível em: https://estatisticafacil.org/glossario/o-que-e-grafos-estrutura-e-aplicacoes/. Acesso em: 08 set. 2025.
5. CAMPELLO, Ricardo J. G. B. *Introdução aos Grafos: Estruturas de Dados, Definição e Terminologia*. ICMC-USP. Disponível em: http://wiki.icmc.usp.br/images/5/59/Grafos_I.pdf. Acesso em: 08 set. 2025.