Relatório de Implementação: Algoritmos de Caminho Mínimo

Disciplina: MC558 - Projeto e Análise de Algoritmos II

Nome: Tomás Conti Loesch

RA: 224991

1. Introdução

Este relatório apresenta a avaliação experimental de cinco algoritmos desenvolvidos para resolver o problema de caminho mínimo em grafos direcionados, simulando cenários de logística urbana. O objetivo principal foi comparar a precisão e o desempenho de tempo entre abordagens exatas (Dijkstra, Bi-Dijkstra, Programação Linear) e abordagens aproximadas focadas em velocidade (Dial e Bi-Dial).

2. Análise de Precisão (Exatidão vs. Aproximação)

Os algoritmos foram divididos em duas categorias de serviço: **Alta Precisão** (pesos originais `double`) e **Velocidade Extrema** (pesos arredondados para `int`).

Conforme observado nos testes, os algoritmos exatos (**Dijkstra, BiDijkstra e PL**) retornaram consistentemente a distância ótima precisa. Já os algoritmos baseados na otimização de Dial (**DialDijkstra e BiDialDijkstra**) apresentaram divergências esperadas em casos onde o arredondamento dos pesos alterou o custo acumulado.

A tabela abaixo resume os erros encontrados nos testes onde houve divergência:

Caso de Teste	Distância Exata	Distância Dial (Aprox.)	Erro Absoluto	Erro Relativo (%)
arq04.in	21.883	21.0	0.883	4.03%
arq05.in	31.388	32.0	0.612	1.95%
arq06.in	48.787	49.0	0.213	0.43%
arq10.in	943.51	944.0	0.490	0.05%

Análise: O erro relativo manteve-se baixo (abaixo de 5% no pior caso e 0.05% no caso de maior caminho), validando a premissa do "Serviço de Velocidade Extrema": sacrifica-se uma precisão marginal em troca do uso de estruturas de dados inteiras, o que é aceitável para roteamento dinâmico em tempo real.

3. Análise de Desempenho (Tempo de Execução)

A medição de tempo revelou diferenças drásticas de desempenho, especialmente nos casos de teste maiores (`arq07` a `arq10`).

3.1. Dijkstra Clássico vs. Bidirecional

A otimização bidirecional provou ser extremamente eficaz para este conjunto de dados.

- **Dijkstra Clássico (arq10):** ~8.23 ms
- **Bi-Dijkstra (arq10):** ~0.21 ms

O **Bi-Dijkstra foi aproximadamente 39 vezes mais rápido** que o Dijkstra clássico no maior caso de teste. Isso ocorre porque a busca bidirecional reduz drasticamente o espaço de busca explorado.

3.2. Impacto da Otimização de Dial

Comparando o `DialDijkstra` com o `Dijkstra` clássico:

- Em casos pequenos (`arq01` a `arq06`), os tempos são comparáveis (escala de microsegundos).
- Em casos maiores, o Dial manteve tempos competitivos, mas a versão **BiDialDijkstra** destacou-se como a mais rápida entre todas as aproximações em vários cenários, atingindo **0.30 ms** no `arq10`.

3.3. Programação Linear (Gurobi)

A abordagem por Programação Linear (PL) serviu como base de validação teórica, mas mostrou-se inviável para uso em tempo real.

- Enquanto os algoritmos combinatórios resolveram o `arq10` em menos de 10 ms, o **PL levou cerca de 4982 ms (quase 5 segundos)**.
- Isso representa um tempo de execução cerca de **500 a 20.000 vezes maior** que as soluções combinatórias. O *overhead* da montagem do modelo e do método Simplex/Barreira é excessivo para o problema de caminho mínimo clássico.

4. Tabela Comparativa de Tempos (Casos Críticos)

Abaixo, um resumo dos tempos (em milissegundos) para os casos de teste mais exigentes:

Algoritmo	arq07.in	arq08.in	arq09.in	arq10.in

Dijkstra	5.52 ms	10.04 ms	4.41 ms	8.23 ms
BiDijkstra	**0.26 ms**	**0.23 ms**	**0.11 ms**	**0.21 ms**
Dial	5.55 ms	9.95 ms	6.18 ms	9.60 ms
BiDial	0.39 ms	0.29 ms	0.22 ms	0.30 ms
Prog. Linear	4744.64 ms	6874.34 ms	4564.88 ms	4982.98 ms

5. Conclusão

Os experimentos demonstraram que, para o cenário proposto:

1. A **Busca Bidirecional (BiDijkstra)** é a técnica mais eficiente, oferecendo a melhor relação entre velocidade e precisão exata, sendo dezenas de vezes mais rápida que a busca unidirecional em grafos grandes.
2. A **Otimização de Dial** cumpre seu papel de aproximação com erros mínimos, sendo viável quando o hardware não suporta operações de ponto flutuante eficientes, embora em CPUs modernas a vantagem de tempo puro sobre o Dijkstra clássico tenha sido marginal nestes testes.
3. A **Programação Linear**, embora correta, é computacionalmente proibitiva para roteamento em tempo real, devendo ser reservada para problemas mais complexos onde restrições adicionais (capacidade, janelas de tempo) inviabilizam algoritmos gulosos.