Comparação de Algoritmos de Caminho Mínimo: Dijkstra, Bidirectional Dijkstra e Dial's Algorithm

RA: 260382

Novembro 2025

1 Introdução

Este relatório apresenta uma análise comparativa de quatro implementações de algoritmos para o problema de caminho mínimo em grafos: Dijkstra clássico, Dijkstra bidirecional, algoritmo de Dial e Dijkstra bidirecional com algoritmo de Dial. Todos os algoritmos foram testados com o mesmo conjunto de instâncias e medidos quanto à precisão dos resultados e tempo de execução.

Os algoritmos implementados são:

- **dijkstra.cpp**: Implementação clássica do algoritmo de Dijkstra usando heap binário
- **biDijkstra.cpp**: Dijkstra bidirecional com heap binário
- **dialDijkstra.cpp**: Algoritmo de Dial (Dijkstra com bucket array)
- **biDialDijkstra.cpp**: Dijkstra bidirecional usando algoritmo de Dial

2 Metodologia

2.1 Ambiente de Teste

Os testes foram executados em um ambiente Linux usando compilador g++ com flags `-O2 -std=c++17`. O tempo de execução foi medido utilizando `std::chrono::high_resolution_clock` após a leitura completa dos dados de entrada.

2.2 Conjunto de Dados

Foram utilizadas 10 instâncias de teste (arq01.in até arq10.in) com diferentes características de grafos, variando em tamanho e densidade de arestas.

3 Análise de Precisão

3.1 Algoritmos Exatos vs. Aproximados

A Tabela 1 mostra a comparação entre os algoritmos exatos (que trabalham com pesos de ponto flutuante) e aproximados (que arredondam os pesos para inteiros).

Table 1: Comparação de Precisão dos Algoritmos

Instância	Resultado Esperado	Dijkstra	BiDijkstra	Dial	BiDial
arq01.in	44	OK	OK	OK	OK
arq02.in	43	OK	OK	OK	OK
arq03.in	17	OK	OK	OK	OK
arq04.in	21.883	OK	OK	ERRO (21.0)	ERRO (21.0)
arq05.in	31.388	OK	OK	ERRO (32.0)	ERRO (32.0)
arq06.in	48.787	OK	OK	OK	OK
arq07.in	839	OK	OK	OK	OK
arq08.in	853	OK	OK	OK	OK
arq09.in	794	OK	OK	OK	OK
arq10.in	943.510	OK	OK	OK	OK

Observações importantes:

- Os algoritmos **Dijkstra** e **BiDijkstra** são exatos, operando com pesos de ponto flutuante
- Os algoritmos **Dial** e **BiDial** são aproximados, arredondando pesos para inteiros
- Nas instâncias arq04 e arq05, os algoritmos aproximados produziram resultados fora da margem de erro de 1%
- A perda de precisão ocorre quando o arredondamento de pesos altera significativamente o caminho ótimo

4 Análise de Desempenho

4.1 Tempo de Execução

A Tabela 2 apresenta os tempos médios de execução para cada algoritmo.

Table 2: Tempo de Execução dos Algoritmos (em ms)

Instância	Dijkstra	BiDijkstra	Dial	BiDial
arq01.in	1.231	2.794	0.511	0.894
arq02.in	1.069	2.551	0.607	0.990
arq03.in	2.157	1.949	0.498	1.605
arq04.in	1.832	2.587	0.616	1.491
arq05.in	2.255	4.037	1.291	1.701
arq06.in	3.559	3.885	0.894	0.882
arq07.in	4.462	1.899	2.071	1.307
arq08.in	6.236	2.124	3.231	1.461
arq09.in	3.270	1.922	1.759	1.463
arq10.in	5.054	1.929	2.700	1.446
Média	3.113	2.568	1.418	1.324

4.2 Análise Comparativa de Desempenho

Ranking por velocidade (do mais rápido ao mais lento):

1. **BiDial** (1.324 ms): Mais rápido, combinando busca bidirecional com bucket array
2. **Dial** (1.418 ms): Segundo mais rápido, beneficiando-se do bucket array para pesos pequenos

3. **BiDijkstra** (2.568 ms): Busca bidirecional com heap tradicional
4. **Dijkstra** (3.113 ms): Implementação clássica, mais lenta

Principais observações:

- A **busca bidirecional** reduz significativamente o espaço de busca, resultando em speedup de até 2x
- O **algoritmo de Dial** é mais eficiente para grafos com pesos pequenos, eliminando operações logarítmicas do heap
- A combinação BiDial oferece o melhor desempenho, sendo aproximadamente **2.35x mais rápido** que o Dijkstra clássico
- O overhead da busca bidirecional é compensado pela redução do espaço explorado

5 Conclusões

Este estudo demonstra o trade-off fundamental entre precisão e performance em algoritmos de caminho mínimo:

Para aplicações que requerem precisão máxima:

- **BiDijkstra** oferece a melhor combinação de exatidão e performance
- Speedup de 1.21x em relação ao Dijkstra clássico mantendo precisão total

Para aplicações que toleram aproximação:

- **BiDial** oferece performance superior com speedup de 2.35x
- Adequado quando pesos são naturalmente inteiros ou precisão de 1% é aceitável

A escolha do algoritmo deve considerar as características específicas do problema: para grafos esparsos com pesos fracionários, BiDijkstra é preferível; para grafos densos com pesos inteiros pequenos, BiDial oferece performance ótima.

A implementação bidirecional se mostrou consistentemente superior às versões unidirecionais, validando sua eficácia para o problema de caminho mínimo ponto-a-ponto.