Bidirectional Dijkstra's Algorithm is Instance-Optimal

Bernhard Haeupler[1,2], Richard Hladík[1,2], Václav Rozhoň[1], Robert E. Tarjan[3], Jakub Tětek[1]

[1] INSAIT, Sofia University "St. Kliment Ohridski"
[2] ETH Zürich
[3] Princeton University

SOSA 2025

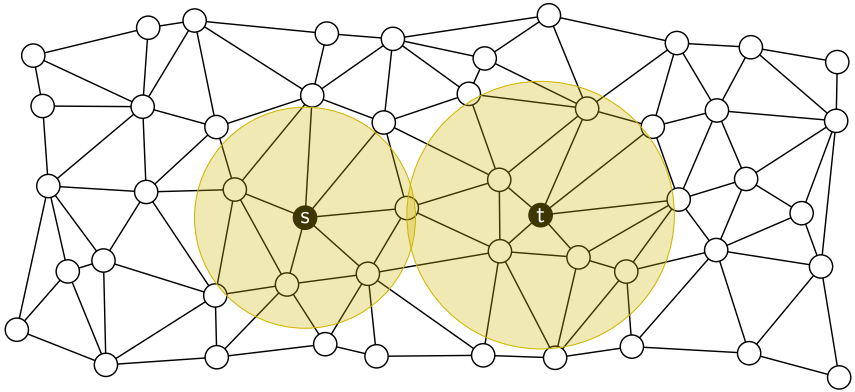
Bidirectional Dijkstra's Algorithm is Instance-Optimal

Bernhard Haeupler[1,2], Richard Hladík[1,2], Václav Rozhoň[1], Robert E. Tarjan[3], Jakub Tětek[1]

[1] INSAIT, Sofia University "St. Kliment Ohridski"
[2] ETH Zürich
[3] Princeton University

SOSA 2025

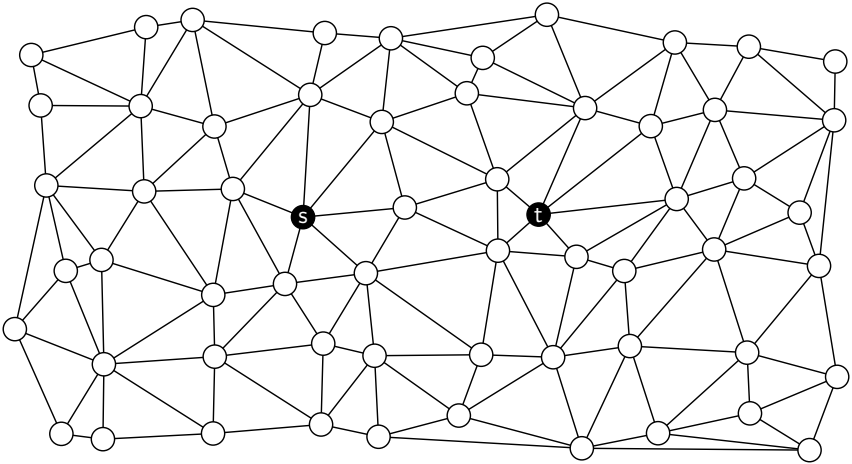
Intro



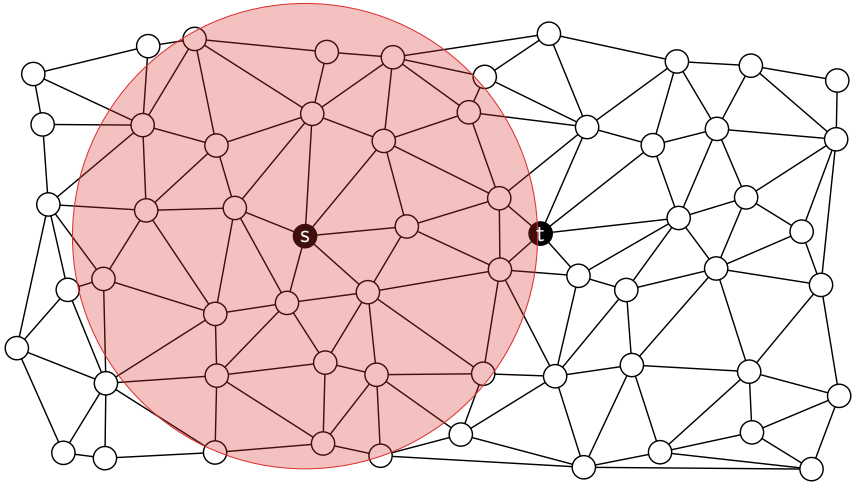
This talk: bidirectional search is optimal on every graph

How quickly can you find the shortest path?

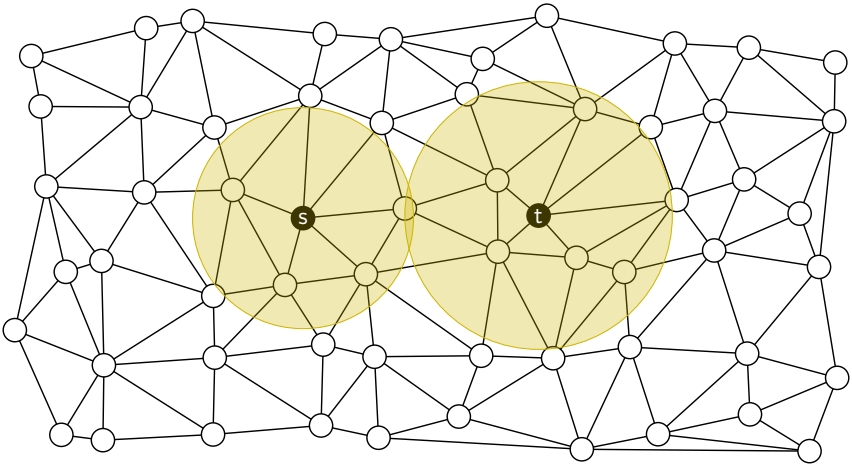
How quickly can you find the shortest path?



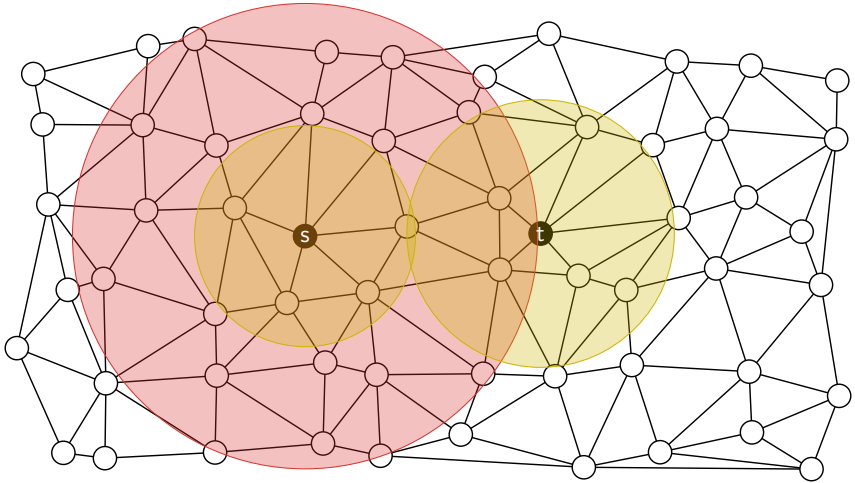
How quickly can you find the shortest path: Dijkstra



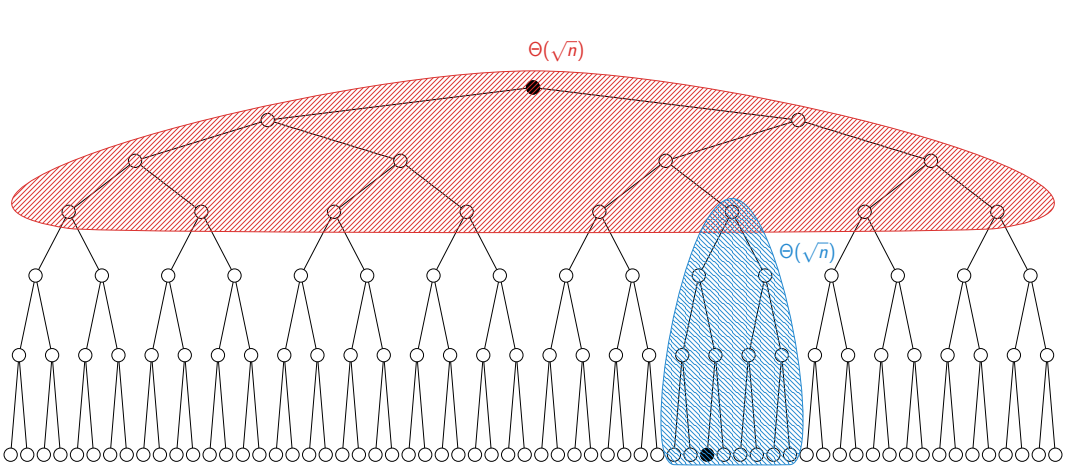
How quickly can you find the shortest path: bidirectional Dijkstra



How quickly can you find the shortest path: bidirectional Dijkstra



Bidirectional search can make a big difference!



Outline

- problem statement
- model
- instance-optimality
- the result
- sketch of the key idea
- related work & debrief

Shortest *st*-path

Input:

Shortest *st*-path

Input:

- multigraph G

Shortest *st*-path

Input:

- multigraph G
- edges have positive lengths $\ell(e)$

Shortest st-path

Input:

- multigraph G
- edges have positive lengths $\ell(e)$
- $s, t \in V(G)$

Shortest st-path

Input:

- multigraph G
- edges have positive lengths $\ell(e)$
- $s, t \in V(G)$

Output: a shortest st-path

Shortest st-path

Input:

- multigraph G
- edges have positive lengths $\ell(e)$
- $s, t \in V(G)$

Output: a shortest st-path

Model: G can be only accessed via queries:

- **Degree**$(v) \to$ degree of v
- **Neighbor**$(v, i) \to$ (vertex id, edge weight) of i-th neighbor

Shortest st-path

Input:

- multigraph G
- edges have positive lengths $\ell(e)$
- $s, t \in V(G)$

Output: a shortest st-path

Model: G can be only accessed via queries:

- **Degree**$(v) \to$ degree of v
- **Neighbor**$(v, i) \to$ (vertex id, edge weight) of i-th neighbor

Goal: minimize *query complexity* ($=$ #queries)

Shortest st-path

Input:

- multigraph G
- edges have positive lengths $\ell(e)$
- $s, t \in V(G)$

Output: a shortest st-path

Model: G can be only accessed via queries:

- **Degree**$(v) \to$ degree of v
- **Neighbor**$(v, i) \to$ (vertex id, edge weight) of i-th neighbor

Goal: minimize *query complexity* ($=$ #queries)
(results also hold for time complexity up to a $\log n$ factor)

Instance Optimality

A is instance-optimal if:

$$\exists c \; \forall \text{correct } A' \; \forall input \quad \text{Queries}_A(input) \leq c \cdot \text{Queries}_{A'}(input)$$

Instance Optimality

A is instance-optimal if:

$$\exists c \; \forall \text{correct } A' \; \forall input \quad \text{Queries}_A(input) \leq c \cdot \text{Queries}_{A'}(input)$$

"No other algorithm is faster even on a single input (up to a constant factor)"

Instance Optimality

A is instance-optimal if:

$$\exists c \; \forall \text{correct } A' \; \forall input \quad \text{Queries}_A(input) \leq c \cdot \text{Queries}_{A'}(input)$$

"No other algorithm is faster even on a single input (up to a constant factor)"

For us, input $= (G, \ell, s, t)$

Instance Optimality

A is instance-optimal if:

$$\exists c \; \forall \text{correct } A' \; \forall input \quad \text{Queries}_A(input) \leq c \cdot \text{Queries}_{A'}(input)$$

"No other algorithm is faster even on a single input (up to a constant factor)"

For us, input $= (G, \ell, s, t)$

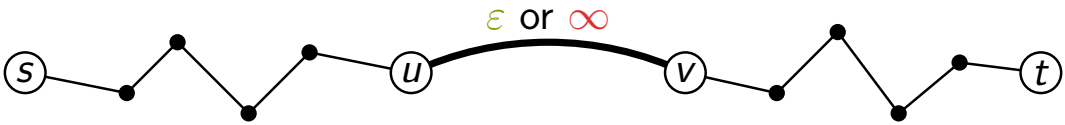
Our result: A variant of bidirectional Dijkstra is instance-optimal (w. r. t. query complexity)

Proof: Key (but insufficient) idea

Claim: Let $uv \in E$ be such that $d(s,u) + d(v,t) < d(s,t)$. Then any correct algorithm needs to see uv.

Proof: Key (but insufficient) idea

Claim: Let $uv \in E$ be such that $d(s,u) + d(v,t) < d(s,t)$. Then any correct algorithm needs to see uv.
Proof: Otherwise it cannot distinguish between $\ell(uv) = \varepsilon$ and $\ell(uv) = \infty$ and is incorrect in at least one case.



Making the proof work

Naive idea:

Making the proof work

Naive idea:

- S = all edges from the previous claim

Making the proof work

Naive idea:

- S = all edges from the previous claim
- prove that bidirectional search only sees S

Making the proof work

Naive idea:

- S = all edges from the previous claim
- prove that bidirectional search only sees S

This fails (bidirectional search sees a lot more edges).

Making the proof work

Naive idea:

- S = all edges from the previous claim
- prove that bidirectional search only sees S

This fails (bidirectional search sees a lot more edges).

. . . but with a stronger claim, we can make it work

Isn't this all trivial?

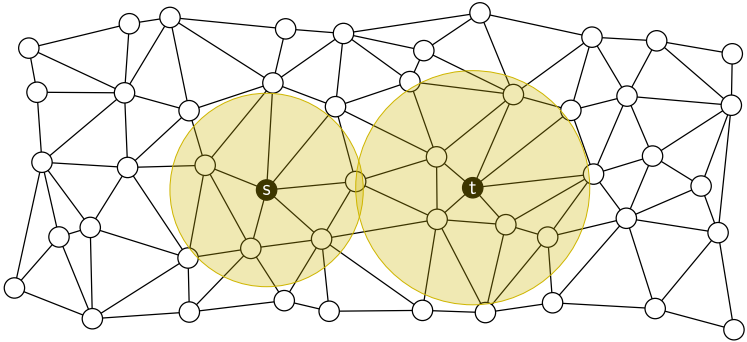
- previous optimality results [Eck+17; Sha+19] assume some knowledge of the graph and are weaker than instance-optimality

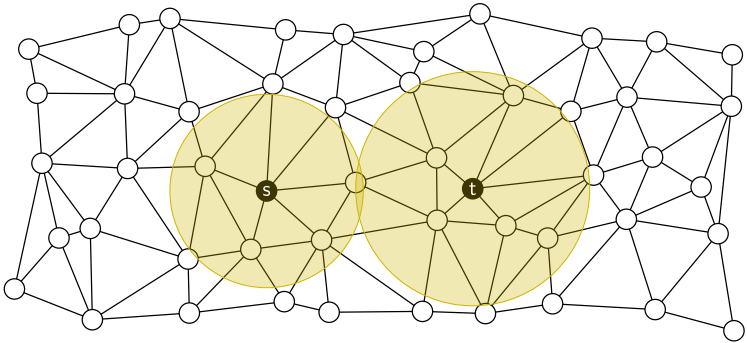
Isn't this all trivial?

- previous optimality results [Eck+17; Sha+19] assume some knowledge of the graph and are weaker than instance-optimality
- bidirectional search has many variants [Dan63; Nic66; Dre69; Poh69], none of the previous variants is instance-optimal
  - we design an instance-optimal variant

Isn't this all trivial?

- previous optimality results [Eck+17; Sha+19] assume some knowledge of the graph and are weaker than instance-optimality
- bidirectional search has many variants [Dan63; Nic66; Dre69; Poh69], none of the previous variants is instance-optimal
  - we design an instance-optimal variant
- we also prove that:
  - allow zero-length edges $\Rightarrow$ instance-optimality impossible
  - **un**weighted graphs $\Rightarrow$ instance-optimality only up to a max-degree factor


s
t



Thank you!

Bibliography I

[Dan63] George B. Dantzig. *Linear Programming and Extensions*. Princeton: Princeton University Press, 1963. ISBN: 9781400884179. DOI: doi:10.1515/9781400884179. URL: https://doi.org/10.1515/9781400884179.

[Nic66] T Alastair J Nicholson. "Finding the shortest route between two points in a network". In: *The computer journal* 9.3 (1966), pp. 275–280.

[Dre69] Stuart E Dreyfus. "An appraisal of some shortest-path algorithms". In: *Operations research* 17.3 (1969), pp. 395–412.

[Poh69] Ira Pohl. *Bi-directional and heuristic search in path problems*. Tech. rep. SLAC National Accelerator Laboratory (SLAC), Menlo Park, CA (United States), 1969.

[Eck+17] Jürgen Eckerle et al. "Sufficient conditions for node expansion in bidirectional heuristic search". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 27. 2017, pp. 79–87.

[Sha+19] Eshed Shaham et al. "Optimally Efficient Bidirectional Search.". In: *IJCAI*. 2019, pp. 6221–6225.

Backup Slides

With zero-length edges, instance-optimality is impossible

Intuition:

With zero-length edges, instance-optimality is impossible

Intuition:

- fix G, s, t

With zero-length edges, instance-optimality is impossible

Intuition:

- fix G, s, t
- set all lengths to 0

With zero-length edges, instance-optimality is impossible

Intuition:

- fix G, s, t
- set all lengths to 0
- now finding the shortest st-path is equivalent to finding *any* st-path

With zero-length edges, instance-optimality is impossible

Intuition:

- fix G, s, t
- set all lengths to 0
- now finding the shortest st-path is equivalent to finding *any* st-path
- there is a set of trivial dumb algorithms that guess what the path is

Banning 0-length edges helps because when a dumb algorithm finds a path, it still has to check if there isn't some shorter one.

The Bidirectional Search Meta-Algorithm

while the *stopping condition* is not met:
  use a *selection rule* to pick execution $\in \{\text{forward}, \text{backward}\}$
  do some work in the given execution

The Bidirectional Search Meta-Algorithm

while the *stopping condition* is not met:
  use a *selection rule* to pick execution $\in \{\text{forward}, \text{backward}\}$
  do some work in the given execution

- **Stopping condition:**

The Bidirectional Search Meta-Algorithm

while the *stopping condition* is not met:
  use a *selection rule* to pick execution $\in \{\text{forward}, \text{backward}\}$
  do some work in the given execution

- **Stopping condition:**
  - executions meet for the first time [Dre69]

The Bidirectional Search Meta-Algorithm

while the *stopping condition* is not met:
  use a *selection rule* to pick execution $\in \{\text{forward}, \text{backward}\}$
  do some work in the given execution

- **Stopping condition:**
  - executions meet for the first time [Dre69]
  - maintain a distance bound on the shortest path, stop when exceeded [Poh69]

The Bidirectional Search Meta-Algorithm

while the *stopping condition* is not met:
  use a *selection rule* to pick execution $\in \{\text{forward}, \text{backward}\}$
  do some work in the given execution

- **Stopping condition:**
  - executions meet for the first time [Dre69]
  - maintain a distance bound on the shortest path, stop when exceeded [Poh69]
- **Selection rule:**

The Bidirectional Search Meta-Algorithm

while the *stopping condition* is not met:
  use a *selection rule* to pick execution $\in \{\text{forward}, \text{backward}\}$
  do some work in the given execution

- **Stopping condition:**
  - executions meet for the first time [Dre69]
  - maintain a distance bound on the shortest path, stop when exceeded [Poh69]
- **Selection rule:**
  - explore one vertex at a time according to some rule [Dan63; Nic66; Poh69]

The Bidirectional Search Meta-Algorithm

while the *stopping condition* is not met:
  use a *selection rule* to pick execution $\in \{\text{forward}, \text{backward}\}$
  do some work in the given execution

- **Stopping condition:**
  - executions meet for the first time [Dre69]
  - maintain a distance bound on the shortest path, stop when exceeded [Poh69]
- **Selection rule:**
  - explore one vertex at a time according to some rule [Dan63; Nic66; Poh69]
    - not instance-optimal!

The Bidirectional Search Meta-Algorithm

while the *stopping condition* is not met:
  use a *selection rule* to pick execution $\in \{\text{forward}, \text{backward}\}$
  do some work in the given execution

- **Stopping condition:**
  - executions meet for the first time [Dre69]
  - maintain a distance bound on the shortest path, stop when exceeded [Poh69]
- **Selection rule:**
  - explore one vertex at a time according to some rule [Dan63; Nic66; Poh69]
    - not instance-optimal!
  - **our rule:** alternate doing $\mathcal{O}(1)$ work in the forward and backward iterations