A lower bound of 4 for online graph exploration

LATIN 2026, Florianópolis

Júlia Baligács



UNIVERSITY OF OXFORD

Funded by the European Union

The online graph exploration problem

task: visit all cities in an unknown area

current position:



roads:





The online graph exploration problem

task: visit all cities in an unknown area

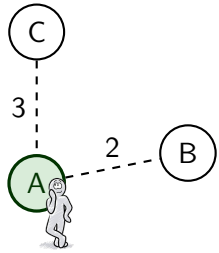
current position:

Avenfeld

roads:

Birkendorf 2km

Cederberg 3km



The online graph exploration problem

task: visit all cities in an unknown area

current position:

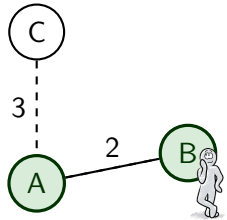
Birkendorf

roads:

Avenfeld 2km

Cederberg 4km

Dachmoor 5km



- tour: $A \xrightarrow{2} B$

The online graph exploration problem

task: visit all cities in an unknown area

current position:

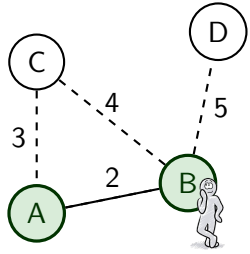
Birkendorf

roads:

Avenfeld 2km

Cederberg 4km

Dachmoor 5km



- tour: $A \xrightarrow{2} B$

The online graph exploration problem

task: visit all cities in an unknown area

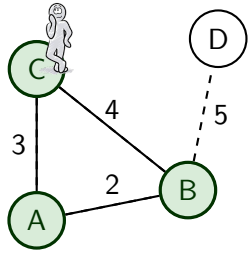
current position:

Cederberg

roads:

Avenfeld 3km

Birkendorf 4km



- tour: $A \xrightarrow{2} B \xrightarrow{4} C$

The online graph exploration problem

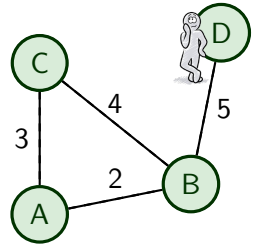
task: visit all cities in an unknown area

current position:

Dachmoor

roads:

Birkendorf 5km



- tour: $A \xrightarrow{2} B \xrightarrow{4} C \xrightarrow{9} D$

The online graph exploration problem

task: visit all cities in an unknown area

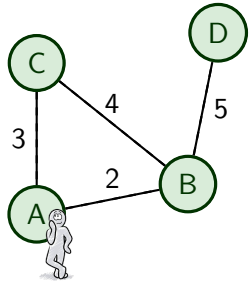
current position:

Avenfeld

roads:

Birkendorf 2km

Cederberg 3km



- tour: $A \xrightarrow{2} B \xrightarrow{4} C \xrightarrow{9} D \xrightarrow{7} A$, total cost: 22

The online graph exploration problem

task: visit all cities in an unknown area

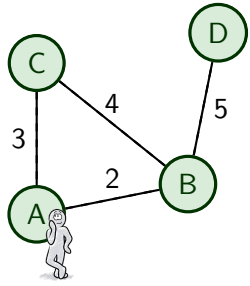
current position:

Avenfeld

roads:

Birkendorf 2km

Cederberg 3km



- tour: $A \xrightarrow{2} B \xrightarrow{4} C \xrightarrow{9} D \xrightarrow{7} A$, total cost: 22
- optimal tour: $A \xrightarrow{2} B \xrightarrow{5} D \xrightarrow{9} C \xrightarrow{3} A$, optimal cost: 19

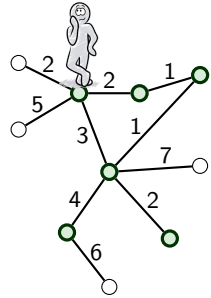
Problem formalization

The online graph exploration problem

setting:

- single agent has to
  - visit every vertex of a graph G (weighted, undirected, connected)
  - return to starting vertex v

objective:



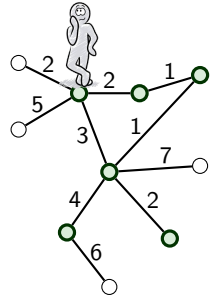
Problem formalization

The online graph exploration problem

setting:

- single agent has to
  - visit every vertex of a graph G (weighted, undirected, connected)
  - return to starting vertex v
- upon visiting new vertex, agent learns:
  - adjacent vertices and weights of corresponding edges

objective:



Problem formalization

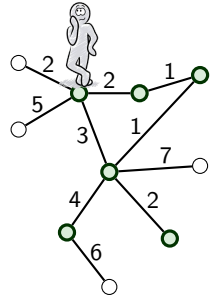
The online graph exploration problem

setting:

- single agent has to
  - visit every vertex of a graph G (weighted, undirected, connected)
  - return to starting vertex v
- upon visiting new vertex, agent learns:
  - adjacent vertices and weights of corresponding edges

objective:

- minimize total walked distance $\textsc{Alg}(G, v)$



Problem formalization

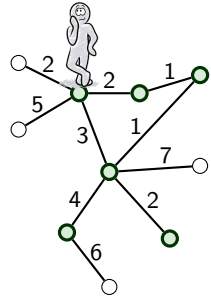
The online graph exploration problem

setting:

- single agent has to
  - visit every vertex of a graph G (weighted, undirected, connected)
  - return to starting vertex v
- upon visiting new vertex, agent learns:
  - adjacent vertices and weights of corresponding edges

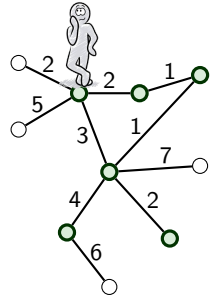
objective:

- minimize total walked distance $\textsc{Alg}(G, v)$



Competitive analysis

Problem formalization



The online graph exploration problem

setting:
- single agent has to
  - visit every vertex of a graph G (weighted, undirected, connected)
  - return to starting vertex v
- upon visiting new vertex, agent learns:
  - adjacent vertices and weights of corresponding edges

objective:
- minimize total walked distance $\textsc{Alg}(G, v)$

Competitive analysis

- $\textsc{Alg}$ is ρ-competitive if $\textsc{Alg}(G, v) \leq \rho \cdot \textsc{Opt}(G)$ for all G, v.

Problem formalization

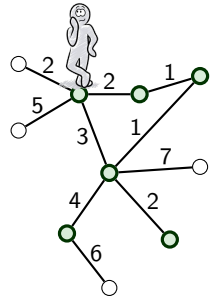
The online graph exploration problem

setting:

- single agent has to
  - visit every vertex of a graph G (weighted, undirected, connected)
  - return to starting vertex v
- upon visiting new vertex, agent learns:
  - adjacent vertices and weights of corresponding edges

objective:

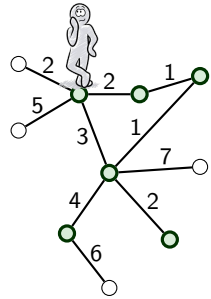
- minimize total walked distance $\textsc{Alg}(G, v)$



Competitive analysis

- $\textsc{Alg}$ is ρ-competitive if $\textsc{Alg}(G, v) \leq \rho \cdot \textsc{Opt}(G)$ for all G, v.
- The competitive ratio of $\textsc{Alg}$ is $\inf\{\rho : \textsc{Alg}$ is ρ-competitive$\}$.

Problem formalization



The online graph exploration problem

setting:

- single agent has to
  - visit every vertex of a graph G (weighted, undirected, connected)
  - return to starting vertex v
- upon visiting new vertex, agent learns:
  - adjacent vertices and weights of corresponding edges

objective:

- minimize total walked distance $\textsc{Alg}(G, v)$

Competitive analysis

- $\textsc{Alg}$ is ρ-competitive if $\textsc{Alg}(G, v) \leq \rho \cdot \textsc{Opt}(G)$ for all G, v.
- The competitive ratio of $\textsc{Alg}$ is $\inf\{\rho : \textsc{Alg}$ is ρ-competitive$\}$.

Open question (Kalyanasundaram, Pruhs, 1994):
Is there a constant-competitive algorithm for online graph exploration?

State of the art and our result

General graphs

- no exploration algorithm with competitive ratio $o(\log n)$ is known yet

State of the art and our result

General graphs

- no exploration algorithm with competitive ratio $o(\log n)$ is known yet
- every algorithm has competitive ratio $\geq 10/3$ [Birx, Disser, Hopp, Karousatou (TCS'21)]

State of the art and our result

General graphs

- no exploration algorithm with competitive ratio $o(\log n)$ is known yet
- every algorithm has competitive ratio $\geq 10/3$ [Birx, Disser, Hopp, Karousatou (TCS'21)]

Restricted graph classes

State of the art and our result

General graphs

- no exploration algorithm with competitive ratio $o(\log n)$ is known yet
- every algorithm has competitive ratio $\geq 10/3$ [Birx, Disser, Hopp, Karousatou (TCS'21)]

Restricted graph classes

- There is a constant-competitive algorithm on
  - planar graphs (16-competitive), [Kalyanasundaram, Pruhs (TCS'94)]



State of the art and our result

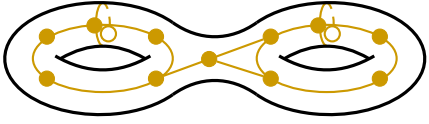
General graphs

- no exploration algorithm with competitive ratio $o(\log n)$ is known yet
- every algorithm has competitive ratio $\geq 10/3$ [Birx, Disser, Hopp, Karousatou (TCS'21)]

Restricted graph classes

- There is a constant-competitive algorithm on
  - planar graphs (16-competitive), [Kalyanasundaram, Pruhs (TCS'94)]
  - bounded genus graphs, [Megow, Mehlhorn, Schweitzer (TCS'12)]





State of the art and our result

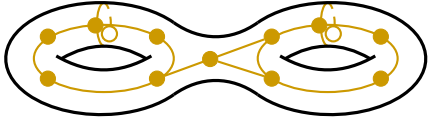
General graphs

- no exploration algorithm with competitive ratio $o(\log n)$ is known yet
- every algorithm has competitive ratio $\geq 10/3$ [Birx, Disser, Hopp, Karousatou (TCS'21)]

Restricted graph classes

- There is a constant-competitive algorithm on
  - planar graphs (16-competitive), [Kalyanasundaram, Pruhs (TCS'94)]
  - bounded genus graphs, [Megow, Mehlhorn, Schweitzer (TCS'12)]
  - graph classes excluding a minor. [B., Disser, Heinrich, Schweitzer (JCSS'26)]





State of the art and our result

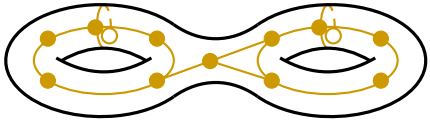
General graphs

- no exploration algorithm with competitive ratio $o(\log n)$ is known yet
- every algorithm has competitive ratio $\geq 10/3$ [Birx, Disser, Hopp, Karousatou (TCS'21)]

Restricted graph classes

- There is a constant-competitive algorithm on
  - planar graphs (16-competitive), [Kalyanasundaram, Pruhs (TCS'94)]
  - bounded genus graphs, [Megow, Mehlhorn, Schweitzer (TCS'12)]
  - graph classes excluding a minor. [B., Disser, Heinrich, Schweitzer (JCSS'26)]





Our result

Theorem. Every algorithm for online graph exploration has competitive ratio at least 4, even on planar graphs.

Warm-up: unweighted graphs

Unweighted: all edges have weight 1

Warm-up: unweighted graphs

Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Warm-up: unweighted graphs

Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof. ▸ cost of DFS: $2(n-1)$

Warm-up: unweighted graphs

Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Warm-up: unweighted graphs

Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Theorem. No algorithm is better than 2-competitive on unweighted graphs.

Warm-up: unweighted graphs

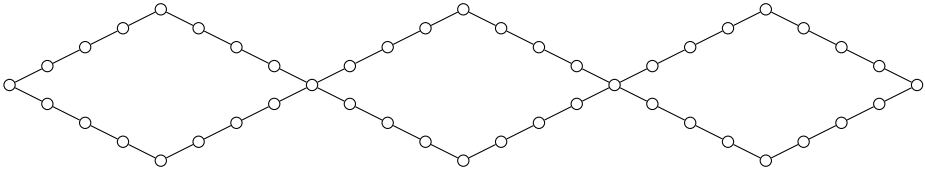
Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Theorem. No algorithm is better than 2-competitive on unweighted graphs.

Proof sketch.



Warm-up: unweighted graphs

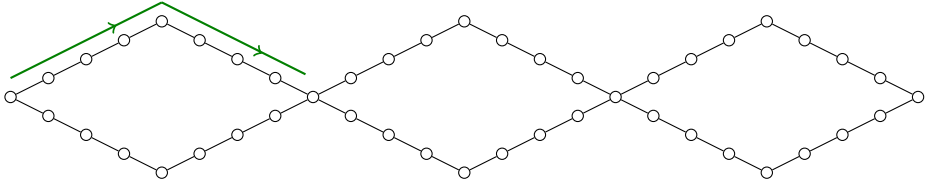
Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Theorem. No algorithm is better than 2-competitive on unweighted graphs.

Proof sketch.



Warm-up: unweighted graphs

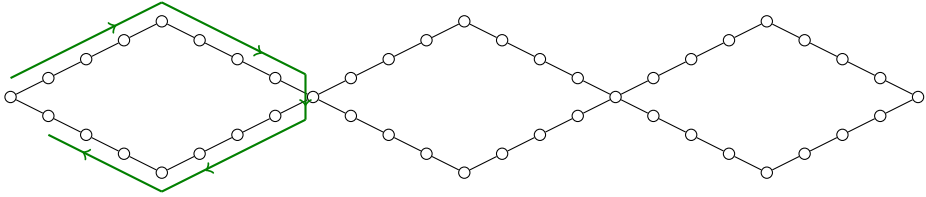
Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Theorem. No algorithm is better than 2-competitive on unweighted graphs.

Proof sketch.



Warm-up: unweighted graphs

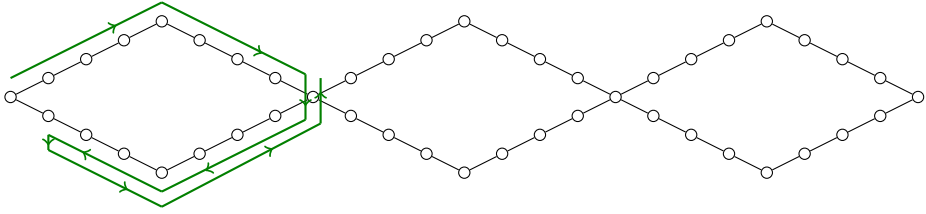
Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Theorem. No algorithm is better than 2-competitive on unweighted graphs.

Proof sketch.



Warm-up: unweighted graphs

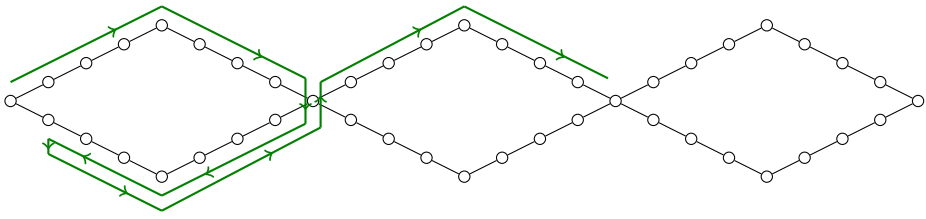
Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Theorem. No algorithm is better than 2-competitive on unweighted graphs.

Proof sketch.



Warm-up: unweighted graphs

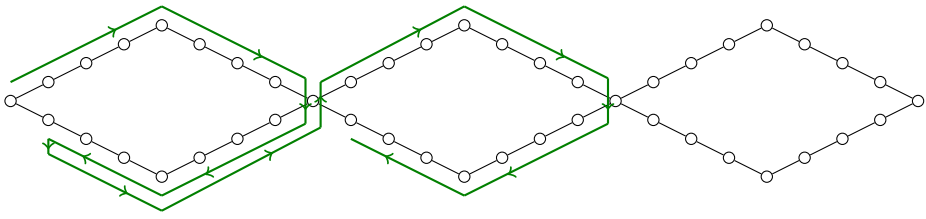
Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Theorem. No algorithm is better than 2-competitive on unweighted graphs.

Proof sketch.



Warm-up: unweighted graphs

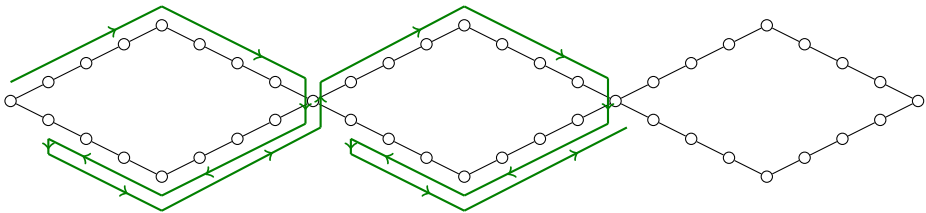
Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Theorem. No algorithm is better than 2-competitive on unweighted graphs.

Proof sketch.



Warm-up: unweighted graphs

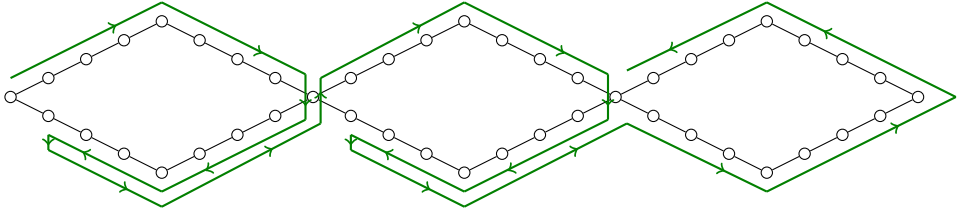
Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Theorem. No algorithm is better than 2-competitive on unweighted graphs.

Proof sketch.



Warm-up: unweighted graphs

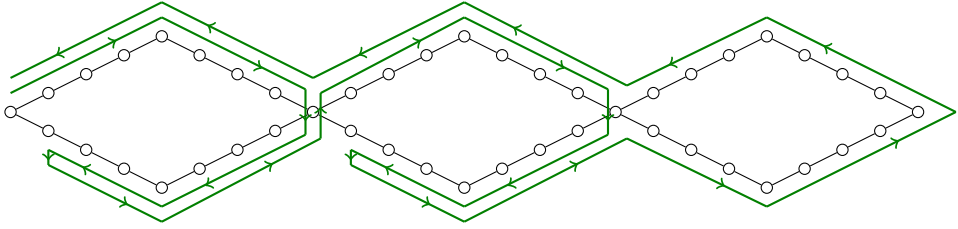
Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Theorem. No algorithm is better than 2-competitive on unweighted graphs.

Proof sketch.



Warm-up: unweighted graphs

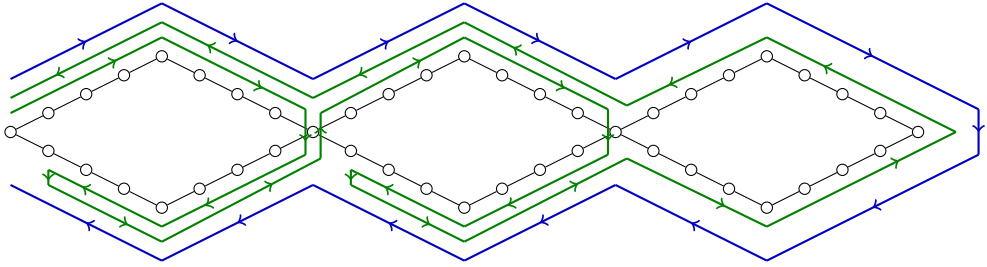
Unweighted: all edges have weight 1

Theorem. DFS is 2-competitive on unweighted graphs.

Proof.
- cost of DFS: $2(n-1)$
- cost of OPT: $\geq n$

Theorem. No algorithm is better than 2-competitive on unweighted graphs.

Proof sketch.

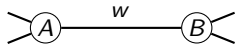


A useful observation

Equivalent setting: agent learns incident edges (instead of neighbors)

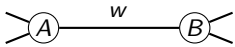
A useful observation

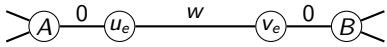
Equivalent setting: agent learns incident edges (instead of neighbors)

Poof sketch: replace 

A useful observation

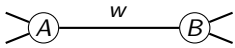
Equivalent setting: agent learns incident edges (instead of neighbors)

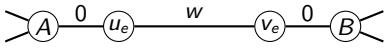
Poof sketch: replace 

by 

A useful observation

Equivalent setting: agent learns incident edges (instead of neighbors)

Poof sketch: replace 

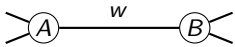
by 

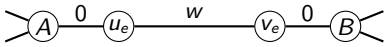
Key consequences

- boundary edge: one unexplored endpoint

A useful observation

Equivalent setting: agent learns incident edges (instead of neighbors)

Poof sketch: replace 

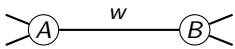
by 

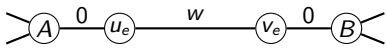
Key consequences

- boundary edge: one unexplored endpoint
- agent cannot tell whether two boundary edges share an unexplored vertex

A useful observation

Equivalent setting: agent learns incident edges (instead of neighbors)

Poof sketch: replace 

by 

Key consequences

- boundary edge: one unexplored endpoint
- agent cannot tell whether two boundary edges share an unexplored vertex
- agent actions have the form: traverse a boundary edge of weight w from vertex v $\rightarrow$ but the adversary chooses which one

Constructing lower bounds

Theorem. Every algorithm for online graph exploration has competitive ratio at least 4, even on planar graphs.

Constructing lower bounds

Theorem. Every algorithm for online graph exploration has competitive ratio at least 4, even on planar graphs.

- the constructed graph will contain bottlenecks (“blocks”) that need to be traversed to reach all vertices

Constructing lower bounds

Theorem. Every algorithm for online graph exploration has competitive ratio at least 4, even on planar graphs.

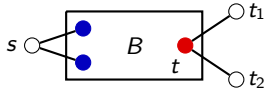
- the constructed graph will contain bottlenecks ("blocks") that need to be traversed to reach all vertices

  → first step: consider "traversal problem"

Block traversal problem

Block traversal problem:

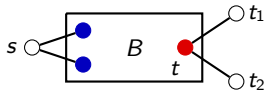
- agent starts in vertex s and must reach one of two targets t_1, t_2
  - $\deg(s) = 2$
  - t_1 and t_2 are adjacent to same vertex t (and no other vertices)



Block traversal problem

Block traversal problem:

- agent starts in vertex s and must reach one of two targets t_1, t_2
  - $\deg(s) = 2$
  - t_1 and t_2 are adjacent to same vertex t (and no other vertices)



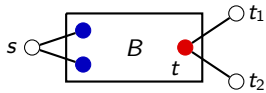
Key values

- $\textsc{Alg}_B$: cost incurred by agent excluding final edge to target

Block traversal problem

Block traversal problem:

- agent starts in vertex s and must reach one of two targets t_1, t_2
  - $\deg(s) = 2$
  - t_1 and t_2 are adjacent to same vertex t (and no other vertices)



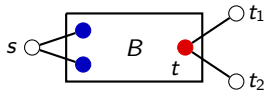
Key values

- $\textsc{Alg}_B$: cost incurred by agent excluding final edge to target
- $\textsc{Opt}_B$: offline optimum cost for starting in s, visiting all vertices of B and ending in t

Block traversal problem

Block traversal problem:

- agent starts in vertex s and must reach one of two targets t_1, t_2
  - $\deg(s) = 2$
  - t_1 and t_2 are adjacent to same vertex t (and no other vertices)



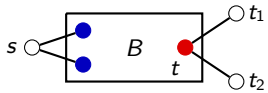
Key values

- $\textsc{Alg}_B$: cost incurred by agent excluding final edge to target
- $\textsc{Opt}_B$: offline optimum cost for starting in s, visiting all vertices of B and ending in t
- l_B: length of a shortest path from s to t

Block traversal problem

Block traversal problem:

- agent starts in vertex s and must reach one of two targets t_1, t_2
  - $\deg(s) = 2$
  - t_1 and t_2 are adjacent to same vertex t (and no other vertices)

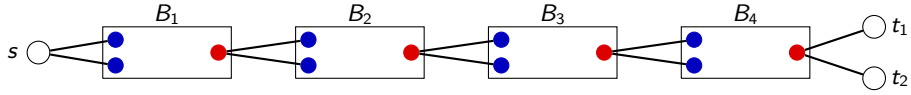


Key values

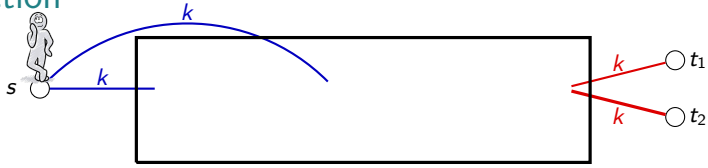
- $\textsc{Alg}_B$: cost incurred by agent excluding final edge to target
- $\textsc{Opt}_B$: offline optimum cost for starting in s, visiting all vertices of B and ending in t
- l_B: length of a shortest path from s to t

Chains of blocks

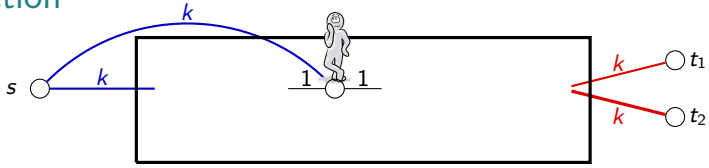
- blocks can be chained: $\textsc{Alg}_B, \textsc{Opt}_B, l_B$ are additive



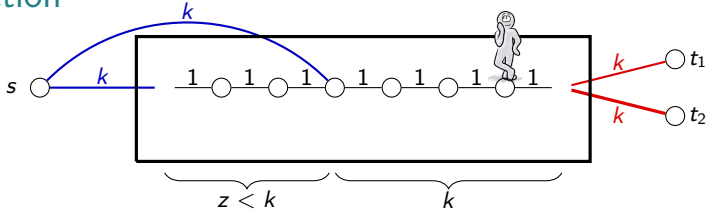
Block construction



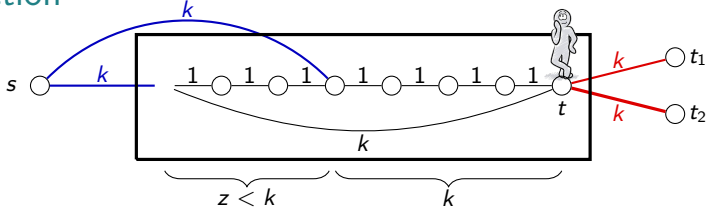
Block construction



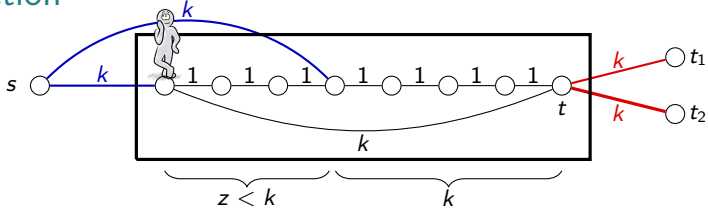
Block construction



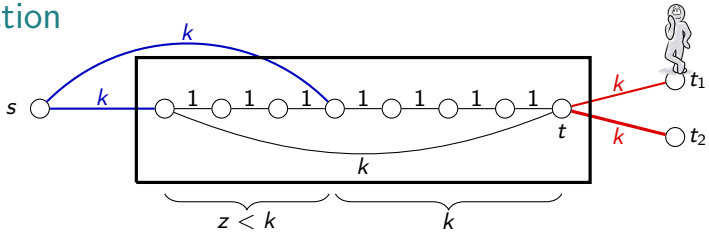
Block construction



Block construction

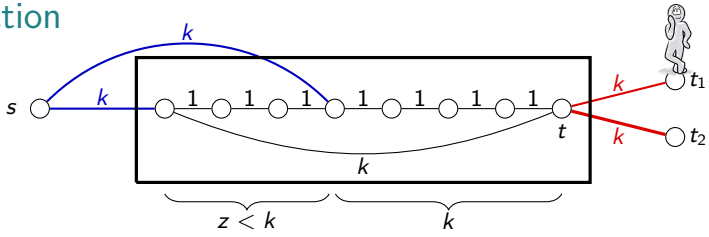


Block construction

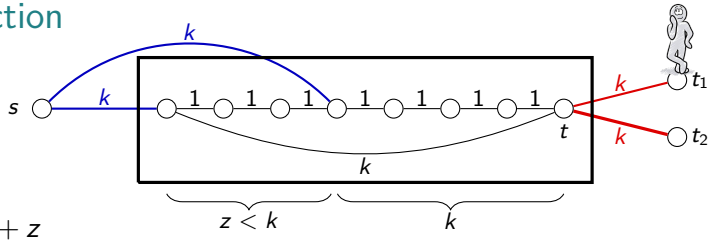


Block construction

- $l_B = 2k$

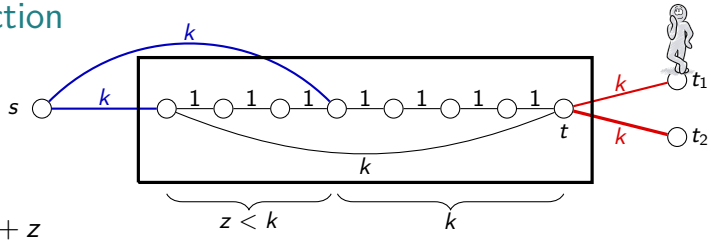


Block construction



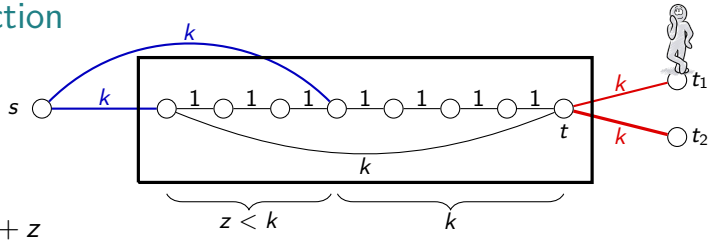
- $l_B = 2k$
- $\mathrm{OPT}_B = 2k + z$

Block construction



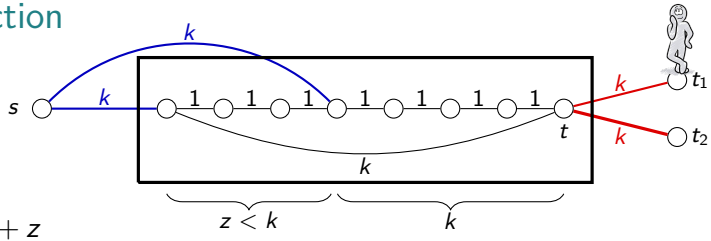
- $l_B = 2k$
- $\textsc{Opt}_B = 2k + z$
- $\textsc{Alg}_B \geq k + 2(z-1) + 3k = 4k + 2(z-1)$

Block construction



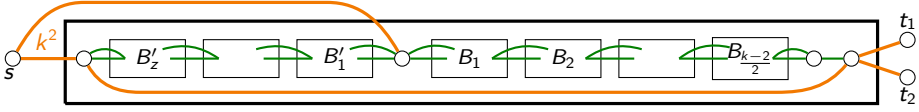
- $l_B = 2k$
- $\text{OPT}_B = 2k + z$
- $\text{ALG}_B \geq k + 2(z-1) + 3k = 4k + 2(z-1)$
- $z_B := \text{OPT}_B - l_B$

Block construction

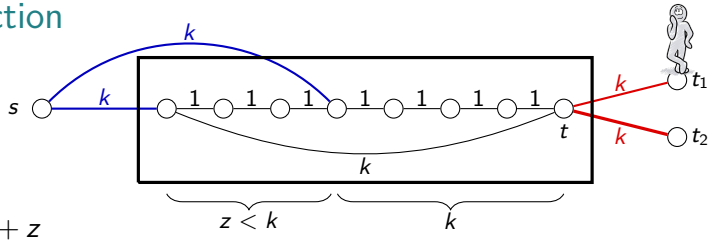


- $l_B = 2k$
- $\mathrm{OPT}_B = 2k + z$
- $\mathrm{ALG}_B \geq k + 2(z-1) + 3k = 4k + 2(z-1)$
- $z_B := \mathrm{OPT}_B - l_B$

key idea: replace edges by same block construction recursively

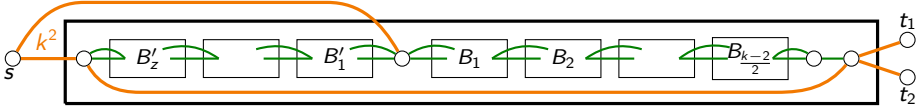


Block construction



- $l_B = 2k$
- $\mathrm{OPT}_B = 2k + z$
- $\mathrm{ALG}_B \geq k + 2(z-1) + 3k = 4k + 2(z-1)$
- $z_B := \mathrm{OPT}_B - l_B$

key idea: replace edges by same block construction recursively

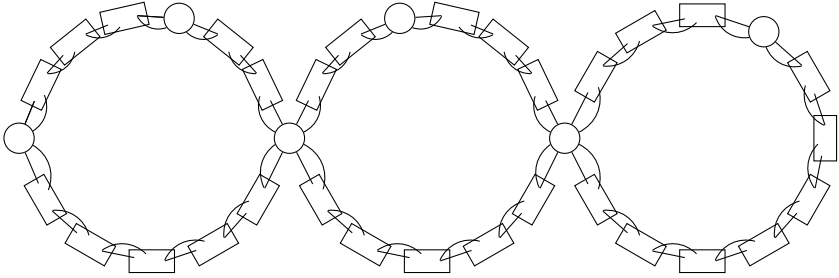


Thm. letting recursion level and $k \to \infty$, we obtain $(\mathrm{ALG}_B - z_B)/\mathrm{OPT}_B \to 3$

Block arrangement

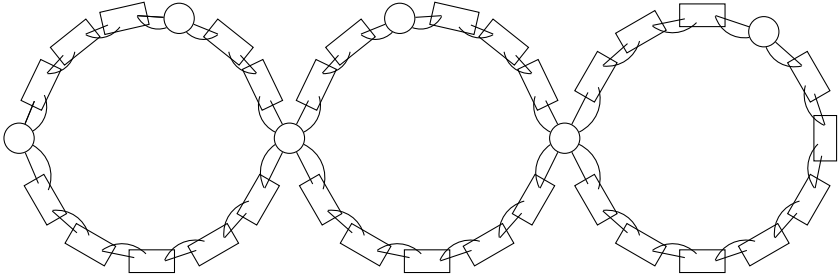
Block arrangement

key idea: arrange blocks as follows



Block arrangement

key idea: arrange blocks as follows

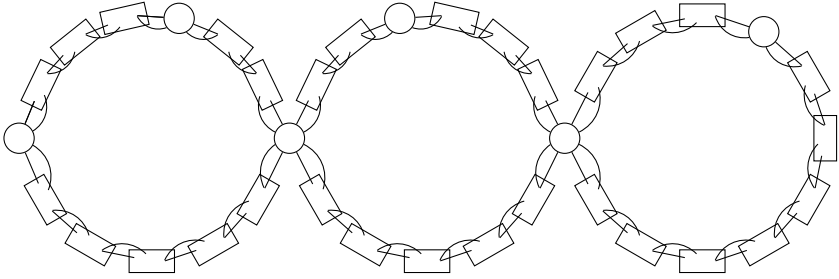


Summary

- ALG traverses (almost) every block twice, OPT traverses every block once

Block arrangement

key idea: arrange blocks as follows

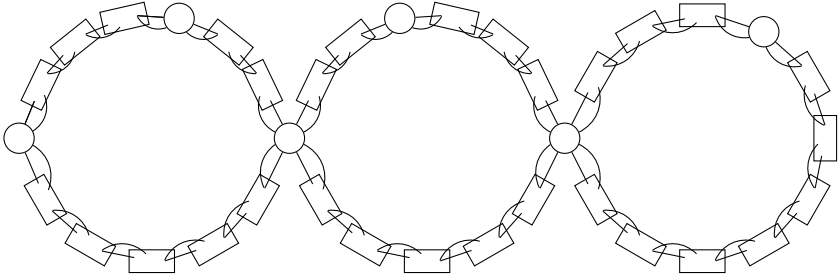


Summary

- $\textsc{Alg}$ traverses (almost) every block twice, $\textsc{Opt}$ traverses every block once
- cost: $\sim \sum_B \textsc{Alg}_B + l_B$

Block arrangement

key idea: arrange blocks as follows

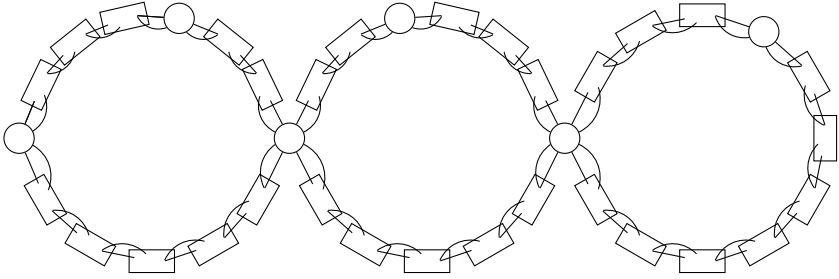


Summary

- $\textsc{Alg}$ traverses (almost) every block twice, $\textsc{Opt}$ traverses every block once
- cost: $\sim \sum_B \textsc{Alg}_B + l_B = \sum_B \textsc{Alg}_B - z_B + \textsc{Opt}_B$

Block arrangement

key idea: arrange blocks as follows

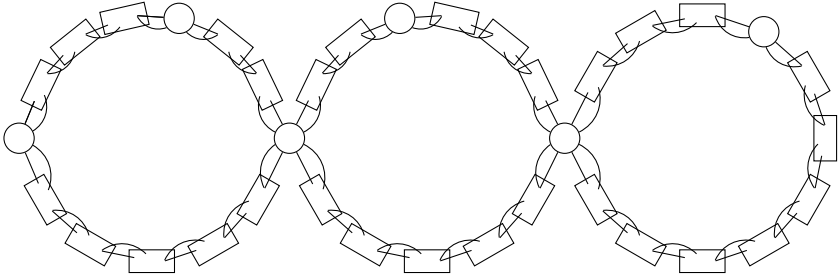


Summary

- $\textsc{Alg}$ traverses (almost) every block twice, $\textsc{Opt}$ traverses every block once
- cost: $\sim \sum_B \textsc{Alg}_B + l_B = \sum_B \textsc{Alg}_B - z_B + \textsc{Opt}_B \geq \sum_B 3\textsc{Opt}_B + \textsc{Opt}_B$

Block arrangement

key idea: arrange blocks as follows

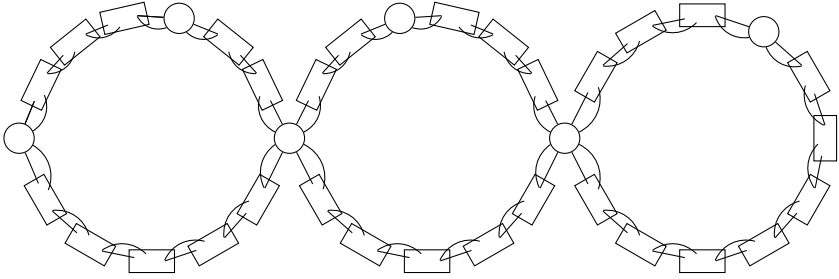


Summary

- $\textsc{Alg}$ traverses (almost) every block twice, $\textsc{Opt}$ traverses every block once
- cost: $\sim \sum_B \textsc{Alg}_B + l_B = \sum_B \textsc{Alg}_B - z_B + \textsc{Opt}_B \geq \sum_B 3\textsc{Opt}_B + \textsc{Opt}_B = 4\textsc{Opt}$

Block arrangement

key idea: arrange blocks as follows



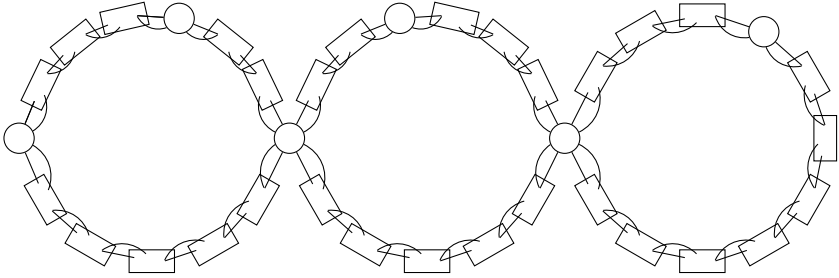
Summary

- $\textsc{Alg}$ traverses (almost) every block twice, $\textsc{Opt}$ traverses every block once
- cost: $\sim \sum_B \textsc{Alg}_B + l_B = \sum_B \textsc{Alg}_B - z_B + \textsc{Opt}_B \geq \sum_B 3\textsc{Opt}_B + \textsc{Opt}_B = 4\textsc{Opt}$

Theorem. Every algorithm for online graph exploration has competitive ratio at least 4, even on planar graphs.

Block arrangement

key idea: arrange blocks as follows





Summary

- ALG traverses (almost) every block twice, OPT traverses every block once
- cost: $\sim \sum_B \mathrm{ALG}_B + l_B = \sum_B \mathrm{ALG}_B - z_B + \mathrm{OPT}_B \geq \sum_B 3\mathrm{OPT}_B + \mathrm{OPT}_B = 4\mathrm{OPT}$

Theorem. Every algorithm for online graph exploration has competitive ratio at least 4, even on planar graphs.