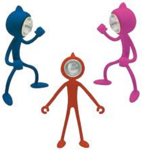
Online Graph Exploration with Advice

Stefan Dobrev[1] and Rastislav Královič[2] and Euripides Markou[3]

Slovak Academy of Sciences, Bratislava.
Stefan.Dobrev@savba.sk

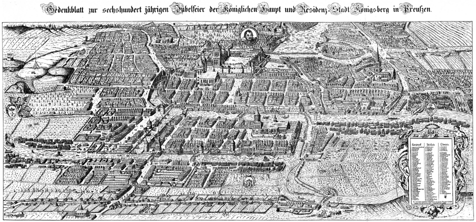
Comenius University, Bratislava.
kralovic@dcs.fmph.uniba.sk

University of Central Greece, Lamia.
emarkou@ucg.gr

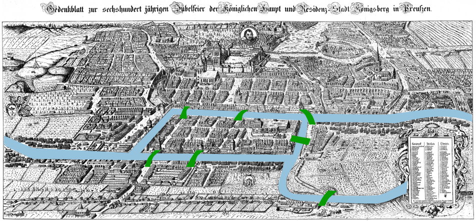


ACAC 2012, Athens

Graph exploration



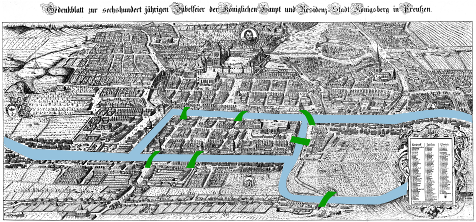
Graph exploration



general statement

Given a … graph,
find a … closed walk that visits all … .

Graph exploration

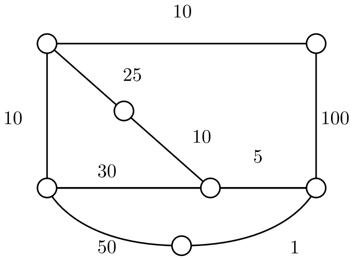


our case

Given a weighted undirected graph,
find a shortest closed walk that visits all vertices.

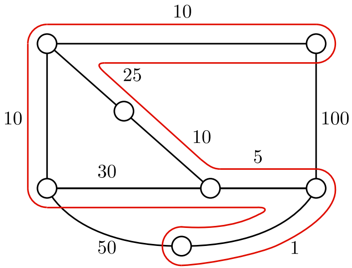
problem statement

Given a weighted undirected graph,
find a shortest closed walk that visits all vertices.



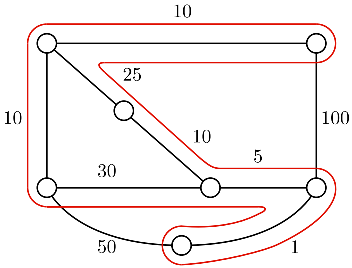
problem statement

Given a weighted undirected graph,
find a shortest closed walk that visits all vertices.



problem statement

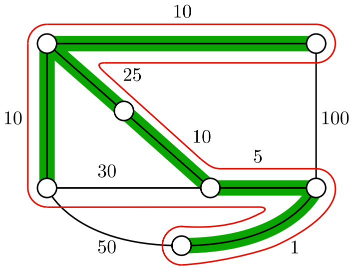
Given a weighted undirected graph,
find a shortest closed walk that visits all vertices.



- equivalent to TSP in the metric closure (vertices may repeat)

problem statement

Given a weighted undirected graph,
find a shortest closed walk that visits all vertices.



- equivalent to TSP in the metric closure (vertices may repeat)
- 2·MST is 2-approximation

online setting: agent

online setting: agent

← can move

online setting: agent

← can not write

← can move

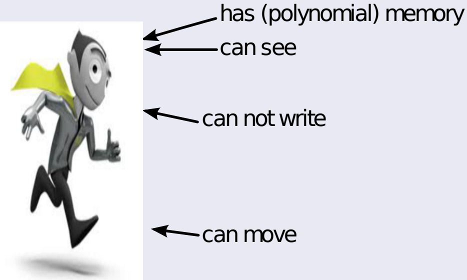
online setting: agent

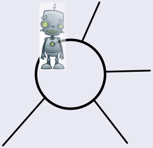
has (polynomial) memory

can not write

can move

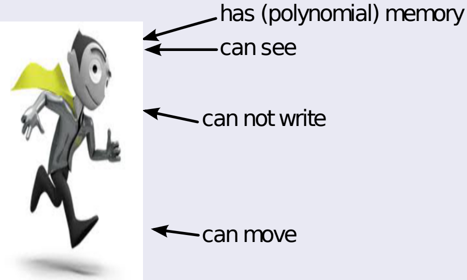
online setting: agent

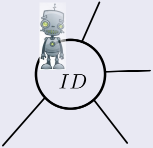




agent in vertex v sees

online setting: agent





agent in vertex v sees

- unique ID

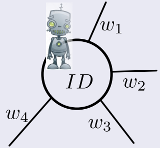
online setting: agent

has (polynomial) memory

can see

can not write

can move

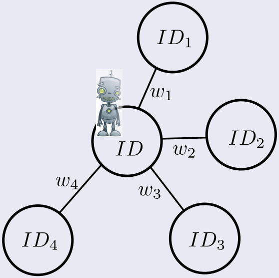


agent in vertex v sees

- unique ID
- weights

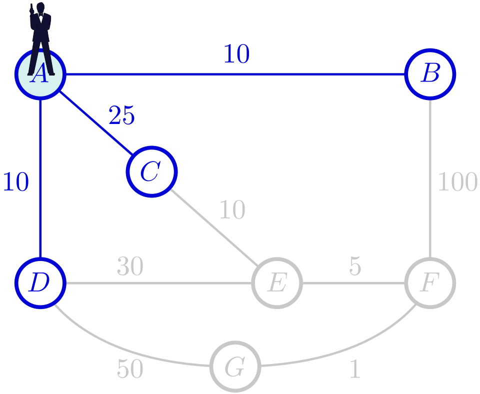
online setting: agent

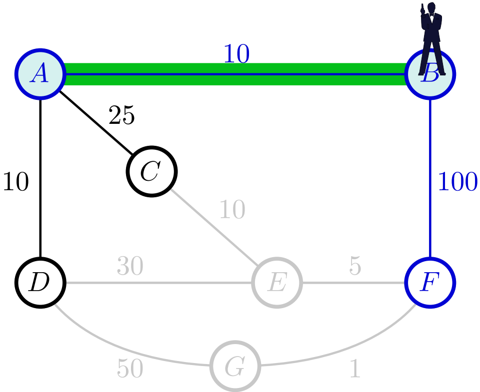
- has (polynomial) memory
- can see
- can not write
- can move



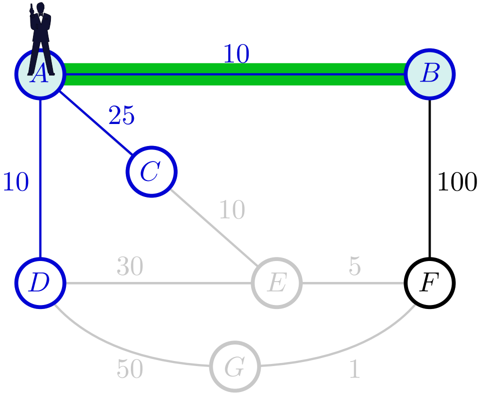
agent in vertex v sees

- unique ID
- weights
- neighbors' IDs

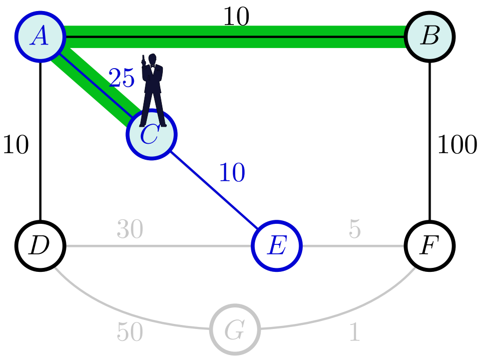

A
B
C
D
E
F
G
10
25
10
10
100
30
5
50
1



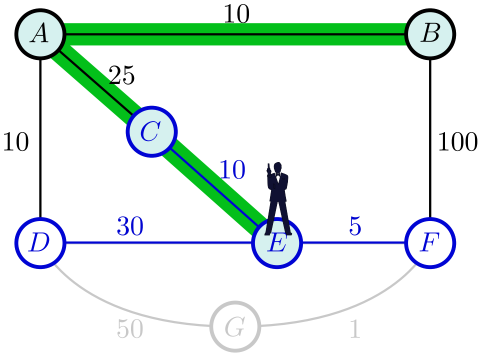
traversed 10



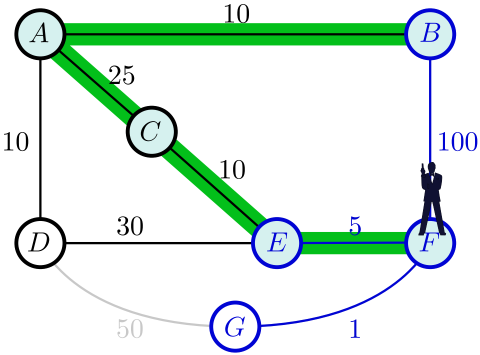
traversed 20



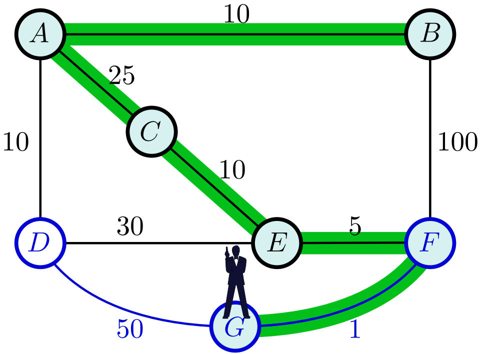
traversed 45



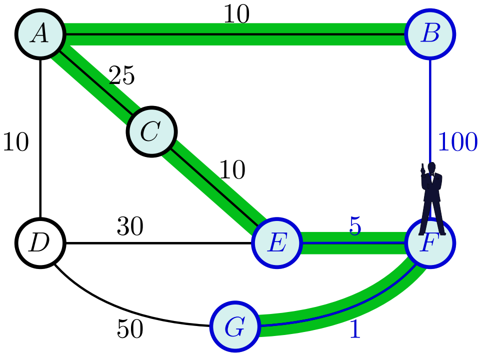
traversed 55



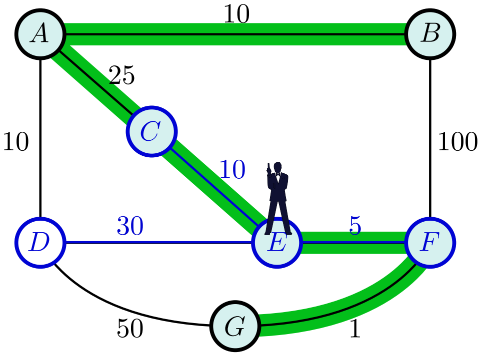
traversed 60



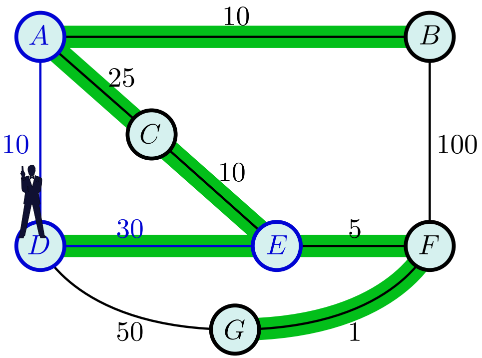
traversed 61



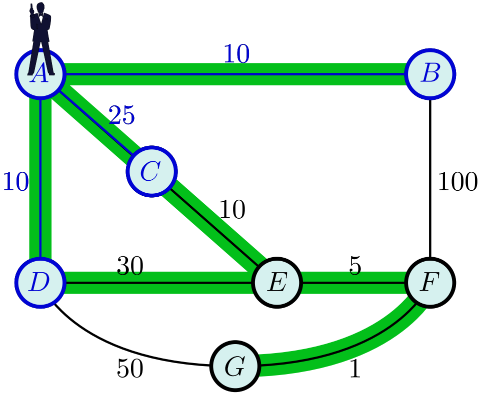
traversed 62



traversed 67



traversed 97



traversed 107

main question

What is the (worst case) **length of** the agent's **traversal compared to** (offline) **optimum?**

Is there a constant competitive algorithm?

main question

What is the (worst case) **length of** the agent's **traversal compared to** (offline) **optimum?**

Is there a constant competitive algorithm?

what has been known about competitive ratio

- [Rosenkranz et al., 1977] NN: $\Theta(\log n)$ even on unweighted planar
- [Myazaki et al., 2009] cycles: $\frac{1+\sqrt{3}}{2} \approx 1.366$, unweighted graphs: 2
- [Kalyanasundaram et al., 1994] planar: 16-competitive (general?)
- [Megow et al., 2011] K+ algorithm is not constant competitive

  genus g: $16(1+2g)$-competitive

  k distinct weights: $2k$-competitive

first task

Improve the $2 - \varepsilon$ lower bound from [Myazaki et al., 2009]

first task

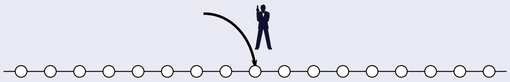
Improve the $2 - \varepsilon$ lower bound from [Myazaki et al., 2009]

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.

we got

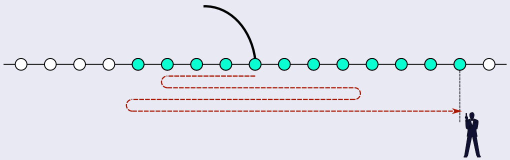
Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.



rough idea

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.

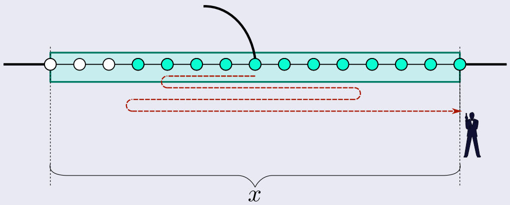


rough idea

- fixed movement

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.

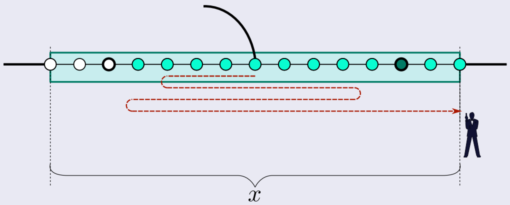


rough idea

- fixed movement
- block of size x

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.

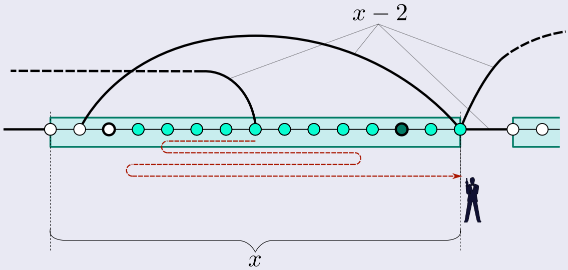


rough idea

- fixed movement
- block of size x
- orientation

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.

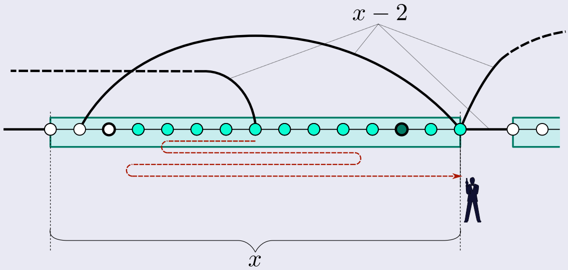


rough idea

- fixed movement
- block of size x
- orientation
- additional edges

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.

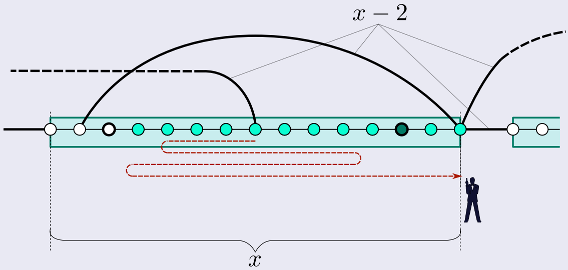


rough idea

- fixed movement
- block of size x
- orientation
- additional edges
- must traverse $\frac{5}{2}(x-1)$

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.



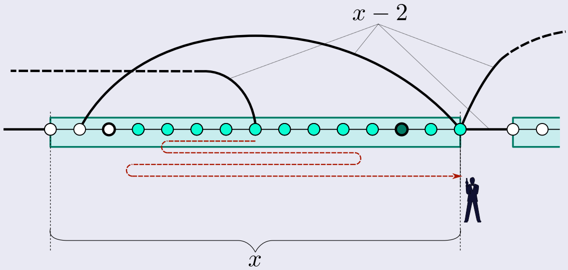
rough idea

- fixed movement
- block of size x
- orientation
- additional edges
- must traverse $\frac{5}{2}(x-1)$

ratio with x blocks: ____________________

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.



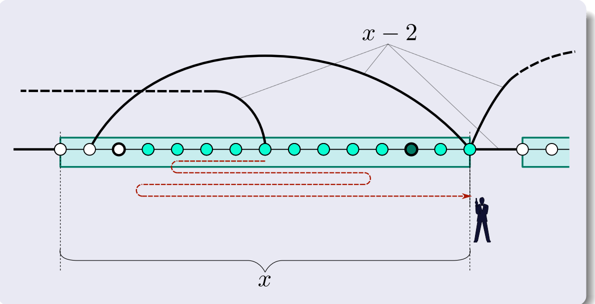
rough idea

- fixed movement
- block of size x
- orientation
- additional edges
- must traverse $\frac{5}{2}(x-1)$

ratio with x blocks: $\frac{\frac{5}{2}(x-1)}{}$

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.



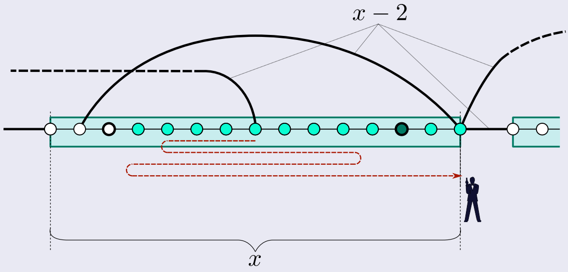
rough idea

- fixed movement
- block of size x
- orientation
- additional edges
- must traverse $\frac{5}{2}(x-1)$

ratio with x blocks: $\frac{(x-2)\frac{5}{2}(x-1)}{}$

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.



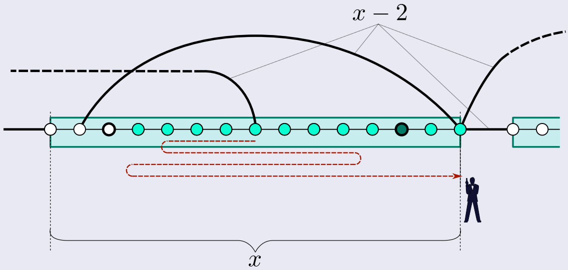
rough idea

- fixed movement
- block of size x
- orientation
- additional edges
- must traverse $\frac{5}{2}(x-1)$

ratio with x blocks: $\frac{(x-2)\frac{5}{2}(x-1)+2(x-1)}{}$

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.



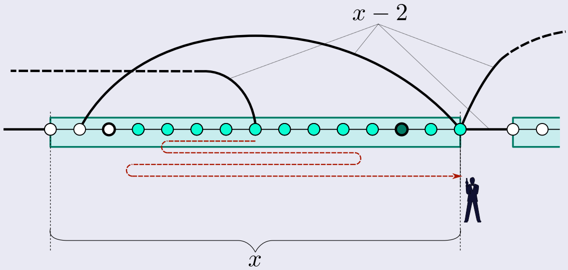
rough idea

- fixed movement
- block of size x
- orientation
- additional edges
- must traverse $\frac{5}{2}(x-1)$

ratio with x blocks: $\frac{(x-2)\frac{5}{2}(x-1)+2(x-1)+x(x-2)}{}$

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.



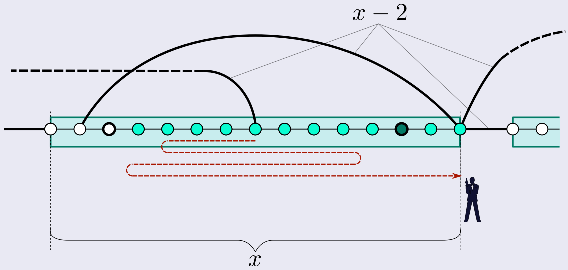
rough idea

- fixed movement
- block of size x
- orientation
- additional edges
- must traverse $\frac{5}{2}(x-1)$

ratio with x blocks: $\frac{(x-2)\frac{5}{2}(x-1)+2(x-1)+x(x-2)}{x(x-1)+x(x-2)}$

we got

Any algorithm $\mathcal{A}$ is at least $\frac{5}{2} - \varepsilon$ competitive on some graph.

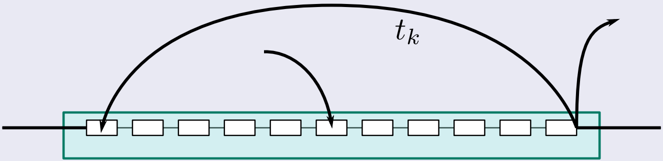


rough idea

- fixed movement
- block of size x
- orientation
- additional edges
- must traverse $\frac{5}{2}(x-1)$

ratio with x blocks: $\frac{(x-2)\frac{5}{2}(x-1)+2(x-1)+x(x-2)}{x(x-1)+x(x-2)} \approx \frac{7}{4}$

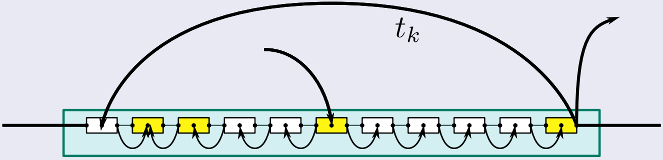
refinement: hierarchical construction



$$
\begin{aligned}
r_1 &= \tfrac{1}{2}(x-1) \\
t_1 &= x-2 \\
o_1 &= x-1 \\
e_1 &\geq \tfrac{5}{2}(x-1) \\
\tilde{e}_1 &\geq x-1
\end{aligned}
$$

ratio x blocks, k levels: $\frac{(x-5)e_k+5\tilde{e}_k+xt_k}{xo_k+xt_k}$

refinement: hierarchical construction



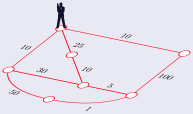
$$
\begin{aligned}
r_1 &= \tfrac{1}{2}(x-1)\\
t_1 &= x-2\\
o_1 &= x-1\\
e_1 &\geq \tfrac{5}{2}(x-1)\\
\tilde{e}_1 &\geq x-1
\end{aligned}
$$

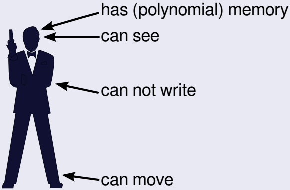
$$
\begin{aligned}
r_{i+1} &= \tfrac{1}{2}(x+1)r_i + \tfrac{1}{2}(x-1)t_i\\
t_{i+1} &= (x-1)t_i + xr_i\\
o_{i+1} &= xo_i + (x-1)t_i\\
e_{i+1} &= (x-4)e_i + 4\tilde{e}_i + \tfrac{5}{2}(x-1)t_i + \tfrac{3x-1}{2}r_i\\
\tilde{e}_{i+1} &\geq (x-5)e_i + 5\tilde{e}_i + (x-1)t_i
\end{aligned}
$$

ratio x blocks, k levels: $\frac{(x-5)e_k+5\tilde{e}_k+xt_k}{xo_k+xt_k} \approx \frac{5}{2} - 2^{-O(\sqrt{\log n})}$

advice complexity

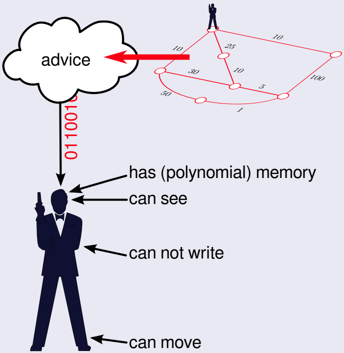
online setting: agent





advice complexity

online setting: agent

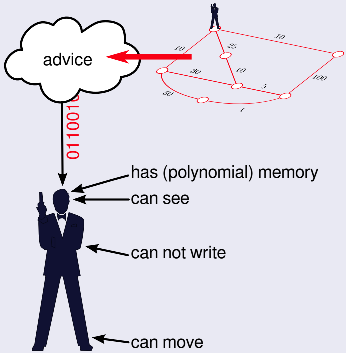


advice

- given to the agent at start
- function of input graph
- s-bit binary string
- "relevant" topology information

advice complexity

online setting: agent



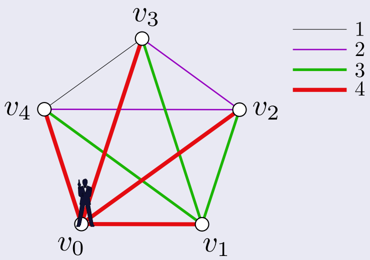
advice

- given to the agent at start
- function of input graph
- s-bit binary string
- "relevant" topology information

advice **size** vs. solution **quality**

lower bound on advice for optimality

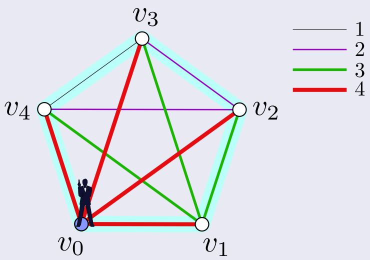
Any optimal algorithm requires $\Omega(n \log n)$ bits in the worst case.



- $w(v_i, v_j) = 4 - \min\{i, j\}$

lower bound on advice for optimality

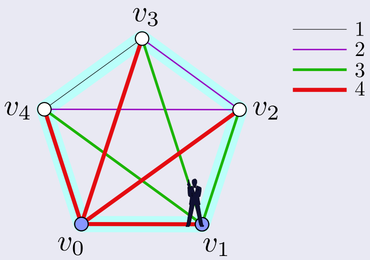
Any optimal algorithm requires $\Omega(n \log n)$ bits in the worst case.



- $w(v_i, v_j) = 4 - \min\{i, j\}$
- unique optimal solution

lower bound on advice for optimality

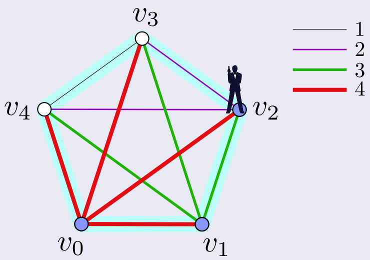
Any optimal algorithm requires $\Omega(n \log n)$ bits in the worst case.



- $w(v_i, v_j) = 4 - \min\{i, j\}$
- unique optimal solution
- agent needs $\log n$ advice

lower bound on advice for optimality

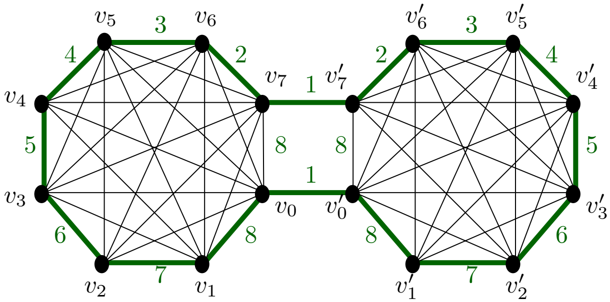
Any optimal algorithm requires $\Omega(n \log n)$ bits in the worst case.



- $w(v_i, v_j) = 4 - \min\{i, j\}$
- unique optimal solution
- agent needs $\log n$ advice
- cannot distinguish next
- problem: reversal

lower bound on advice for optimality

Any optimal algorithm requires $\Omega(n \log n)$ bits in the worst case.



algorithm

There is a constant-competitive algorithm with linear advice.

rough idea

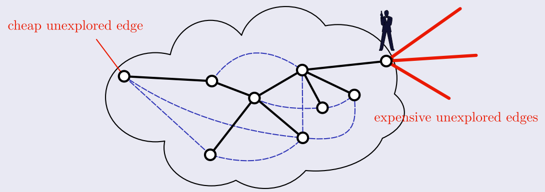
- traverse some tree

algorithm

There is a constant-competitive algorithm with linear advice.

rough idea

- traverse some tree
- DFS has a problem

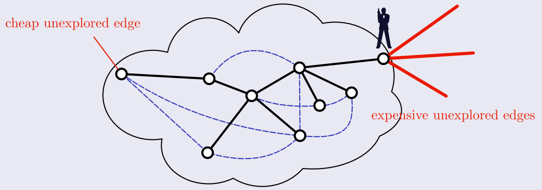


algorithm

There is a constant-competitive algorithm with linear advice.

rough idea

- traverse some tree
- DFS has a problem
- MST of weight M
- *cheap* edge $\leq \frac{M}{n}$



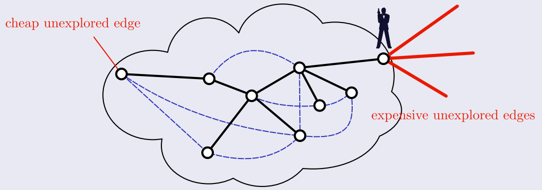
it is always safe to explore a cheap edge

algorithm

There is a constant-competitive algorithm with linear advice.

rough idea

- traverse some tree
- DFS has a problem
- MST of weight M
- *cheap* edge $\leq \frac{M}{n}$



it is always safe to explore a cheap edge

advice tells the value $\lceil \log(\frac{M}{n}) \rceil$ (M is unbounded, but can be done using $O(\log n)$ bits and traversing $O(M)$ total cost.)

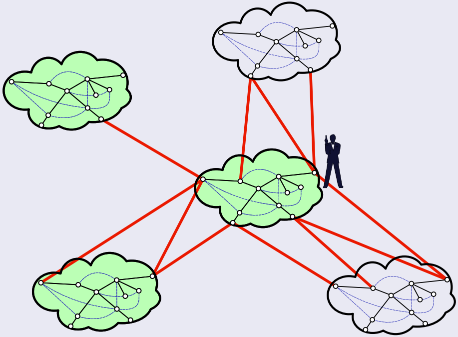
Advice tells the value $\lceil \log(\frac{M}{n}) \rceil$ by encoding: (n, n', p, l')

Search for the first encountered edge e with $w(e) \in [\frac{M}{n^2}, M]$

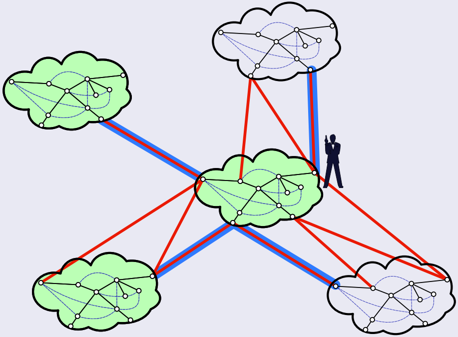
- keep traversing the *cheapest* outgoing edge until n'−th vertex is encountered,
- consider the p−th incident edge e and let $w(e)$ be its weight,
- $\lceil \log(M) \rceil = \lceil \log(w(e)) \rceil + l'$
- (n', p, l') are chosen in such way that e has the right property.

$O(\log n)$ bits are sufficient to encode n, n', p, l'; Cost: $O(M)$

- Such an edge e must exist (otherwise the MST weight $< M$).
- $l' \leq \lceil \log(M) \rceil - \lceil \log(\frac{M}{n^2}) \rceil \leq 2 \log n$
- at most n cheapest $(\frac{M}{n^2})$ outgoing edges; the cost to reach each one is at most $O(\frac{M}{n})$.



- cheap edges always explored by DFS
- *cheap clusters* connected with expensive edges



- cheap edges always explored by DFS
- *cheap clusters* connected with expensive edges
- some expensive edges are *tree edges* (i.e. from MST)
- advice must tell which ones **in an efficient way**

cluster edges

- level 0: cheap
- level i: $w(e) \leq 2^i \frac{M}{n}$
- $\leq \frac{n}{2^i}$ level-i edges in MST

cluster edges

- level 0: cheap
- level i: $w(e) \leq 2^i \frac{M}{n}$
- $\leq \frac{n}{2^i}$ level-i edges in MST

identify tree edges

- levels in parallel
- separate advice for levels
- $O(\log i)$ bits per i-edge

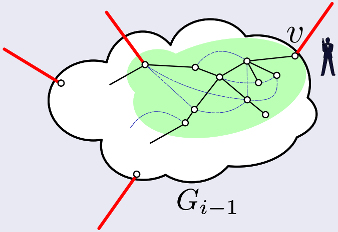
cluster edges

- level 0: cheap
- level i: $w(e) \leq 2^i \frac{M}{n}$
- $\leq \frac{n}{2^i}$ level-i edges in MST

identify tree edges

- levels in parallel
- separate advice for levels
- $O(\log i)$ bits per i-edge

- all i-edges *out*: OUT



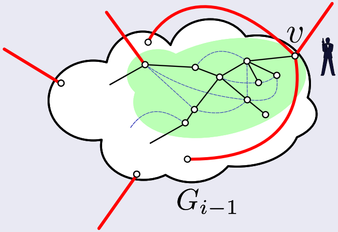
cluster edges

- level 0: cheap
- level i: $w(e) \leq 2^i \frac{M}{n}$
- $\leq \frac{n}{2^i}$ level-i edges in MST

- all i-edges *out*: OUT
- *in* edges: WAIT

identify tree edges

- levels in parallel
- separate advice for levels
- $O(\log i)$ bits per i-edge



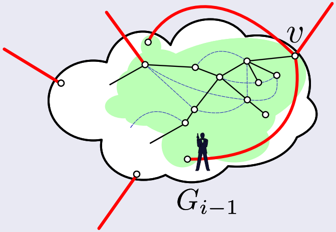
cluster edges

- level 0: cheap
- level i: $w(e) \leq 2^i \frac{M}{n}$
- $\leq \frac{n}{2^i}$ level-i edges in MST

identify tree edges

- levels in parallel
- separate advice for levels
- $O(\log i)$ bits per i-edge

- all i-edges *out*: OUT
- *in* edges: WAIT
- later TRIGGER



cluster edges

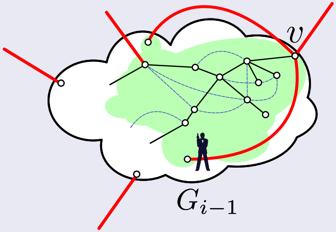
- level 0: cheap
- level i: $w(e) \leq 2^i \frac{M}{n}$
- $\leq \frac{n}{2^i}$ level-i edges in MST

- all i-edges *out*: OUT
- *in* edges: WAIT
- later TRIGGER

managing multiple triggers is the core of the algorithm

identify tree edges

- levels in parallel
- separate advice for levels
- $O(\log i)$ bits per i-edge



open problems

- so is there a constant competitive algorithm or not?
- improve the lower bound $\frac{5}{2} - \varepsilon$
- any general algorithm better than $O(\log n)$?
- what can be done with polylogarithmic advice? or $o(n)$?
- lower bounds on advice / trade-off

open problems

- so is there a constant competitive algorithm or not?
- improve the lower bound $\frac{5}{2} - \varepsilon$
- any general algorithm better than $O(\log n)$?
- what can be done with polylogarithmic advice? or $o(n)$?
- lower bounds on advice / trade-off