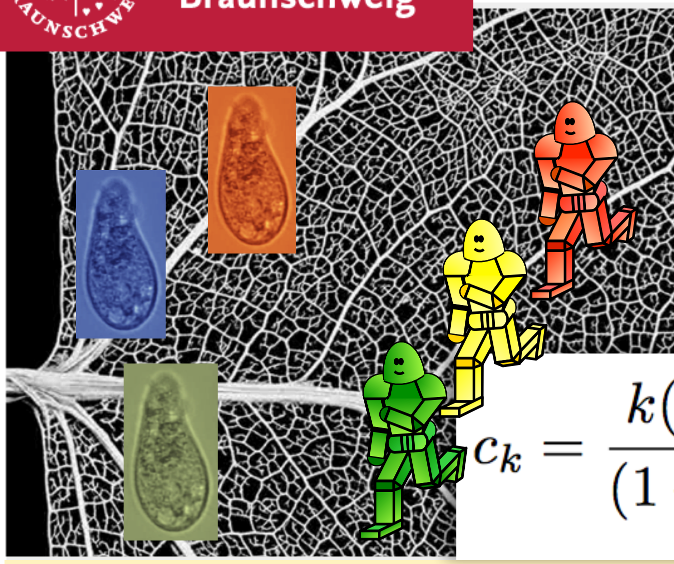
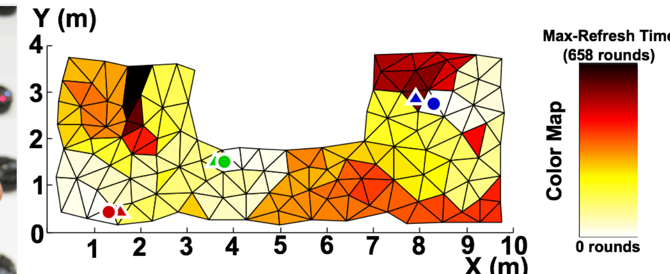
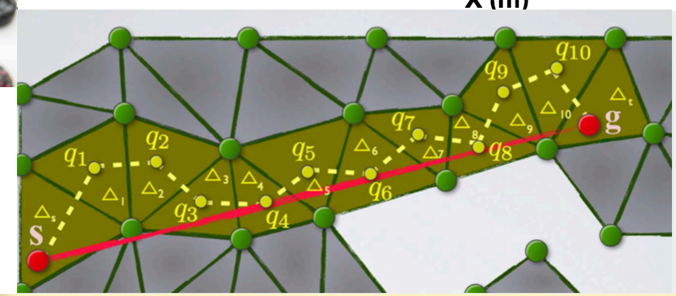
Technische Universität Braunschweig



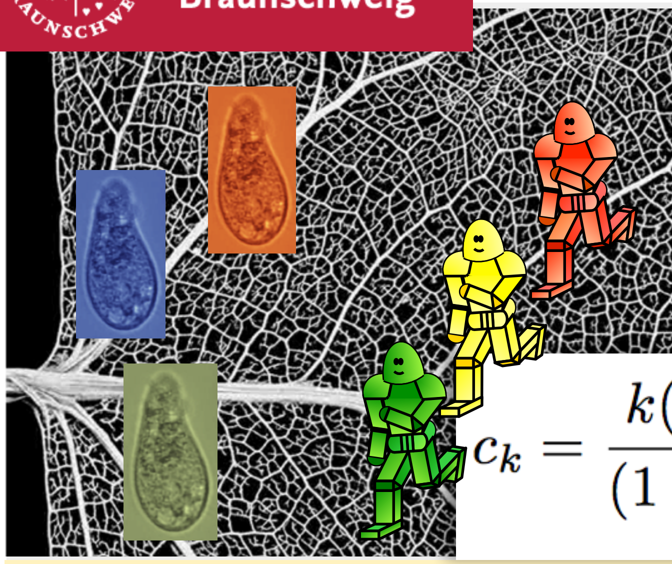


$$c_k = \frac{k(1+\frac{z_k}{k}) - (k-1)}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}}$$

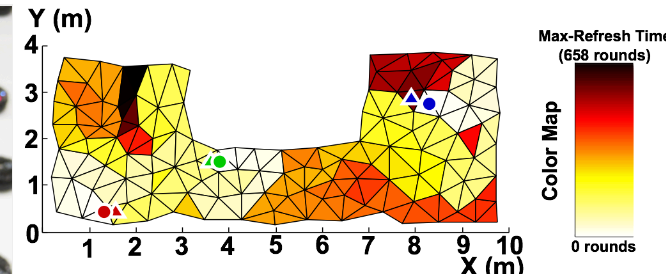


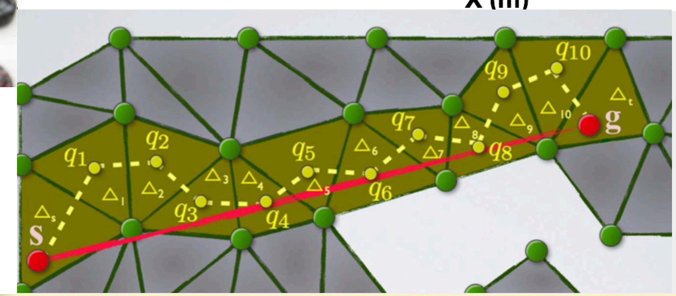
Online Algorithms 2022

Sándor P. Fekete



$$c_k = \frac{k(1+\frac{z_k}{k}) - (k-1)}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}}$$





Online Navigation for Robots

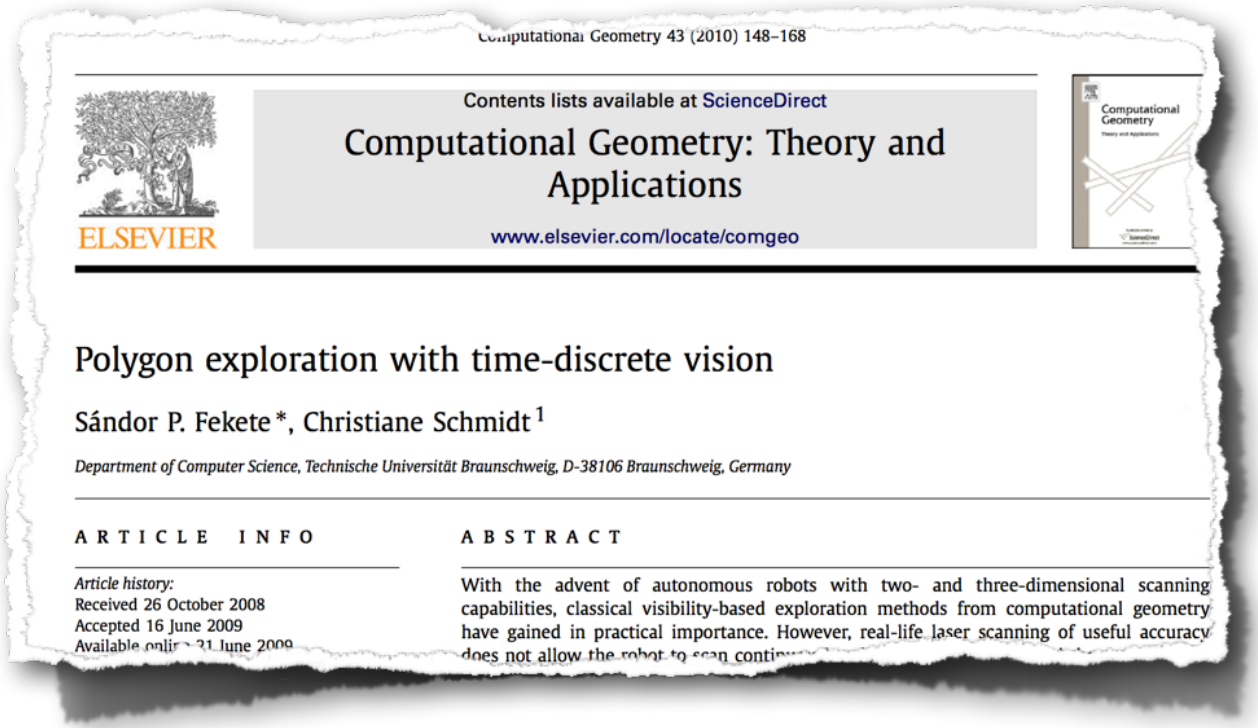
Pravesh Agrawal, Aaron Becker, Erik D. Demaine

Sándor P. Fekete

Golnaz Habibi, Rolf Klein, Alexander Kröller, Andreas Nüchter

Seoung Kyou Lee, James McLurkin, Christiane Schmidt

Part 1.2:
Exploring rectilinear polygons



Computational Geometry 43 (2010) 148–168

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

ELSEVIER

Polygon exploration with time-discrete vision

Sándor P. Fekete *, Christiane Schmidt [1]

Department of Computer Science, Technische Universität Braunschweig, D-38106 Braunschweig, Germany

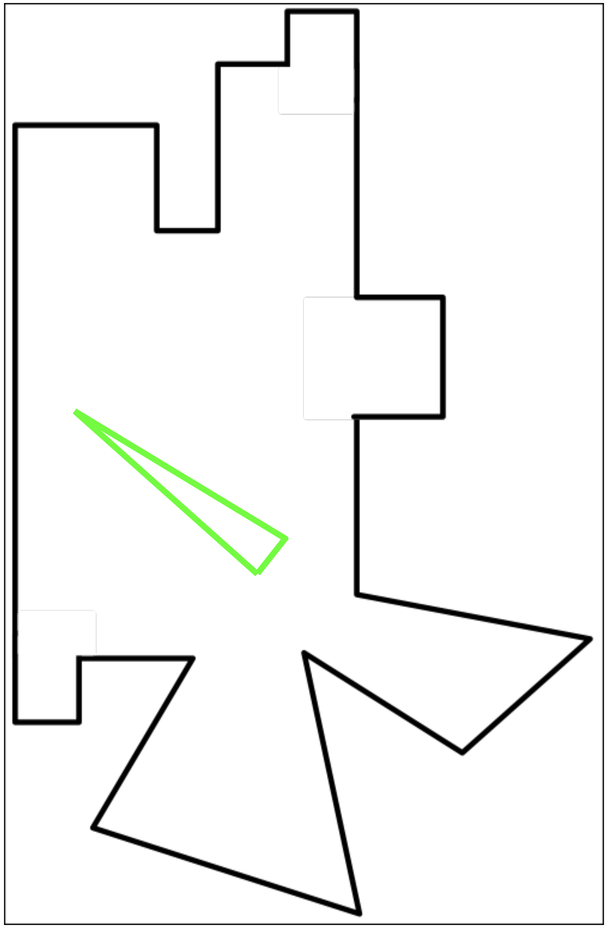
A R T I C L E   I N F O

Article history:
Received 26 October 2008
Accepted 16 June 2009
Available online 21 June 2009

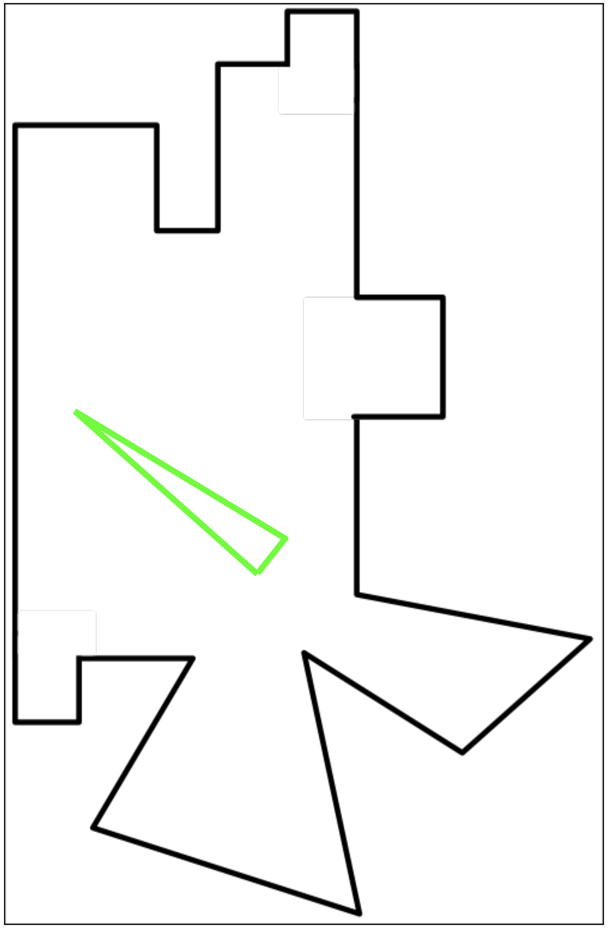
A B S T R A C T

With the advent of autonomous robots with two- and three-dimensional scanning capabilities, classical visibility-based exploration methods from computational geometry have gained in practical importance. However, real-life laser scanning of useful accuracy does not allow the robot to scan conti

Motivation

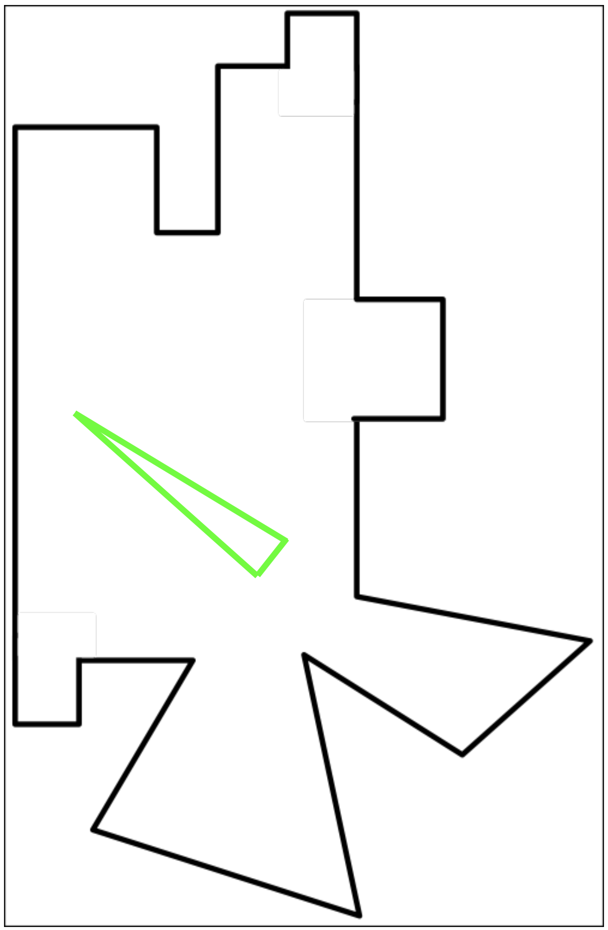


Motivation



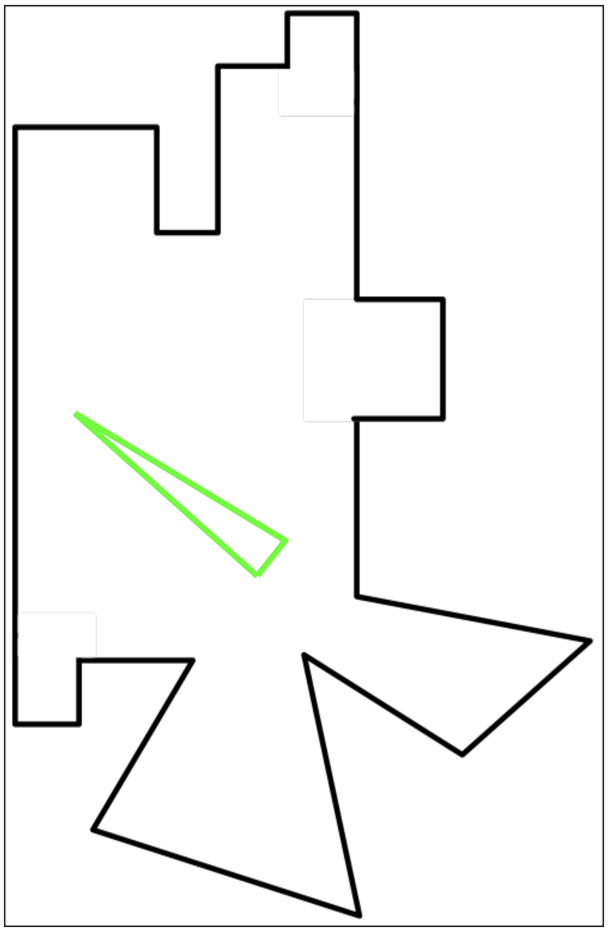
- Watchman problem
- Online, continuous vision:

Motivation



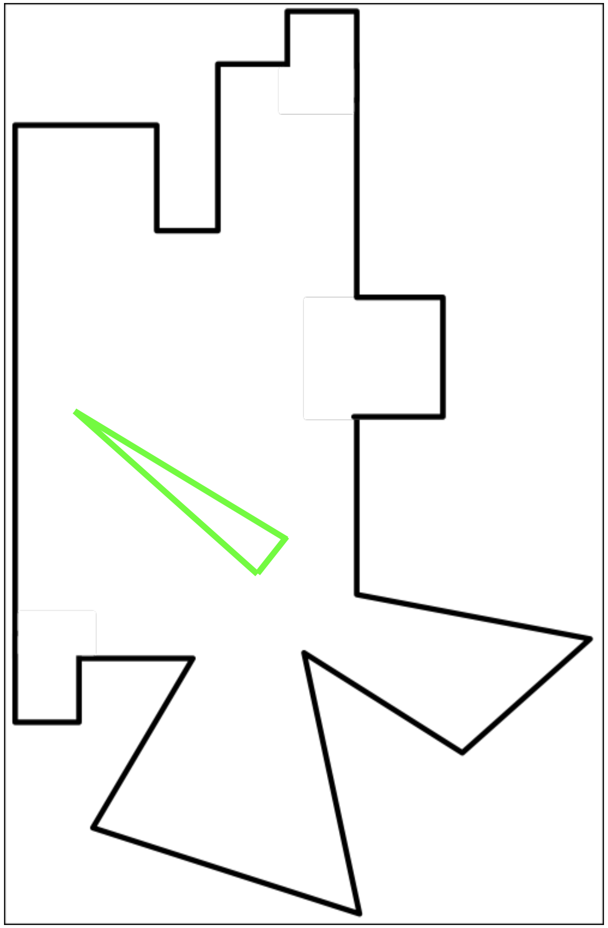
- Watchman problem
- Online, continuous vision:
  - optimum watchman route (L_1-metric) in simple rectilinear polygons (Deng et al.)

Motivation



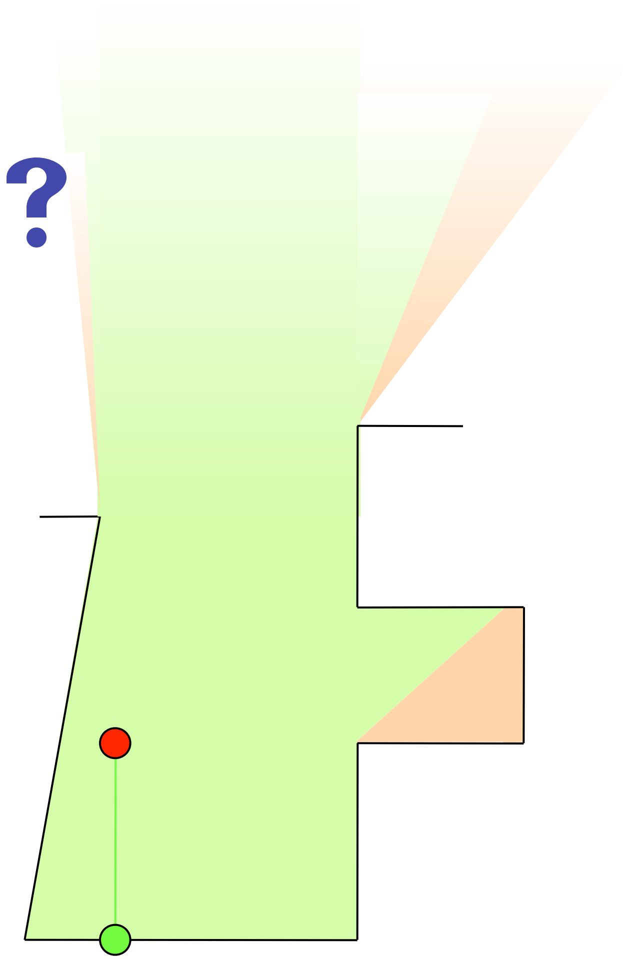
- Watchman problem
- Online, continuous vision:
  - optimum watchman route (L_1-metric) in simple rectilinear polygons (Deng et al.)
  - c=26.5 in simple polygons (Hoffmann et al.)

Motivation

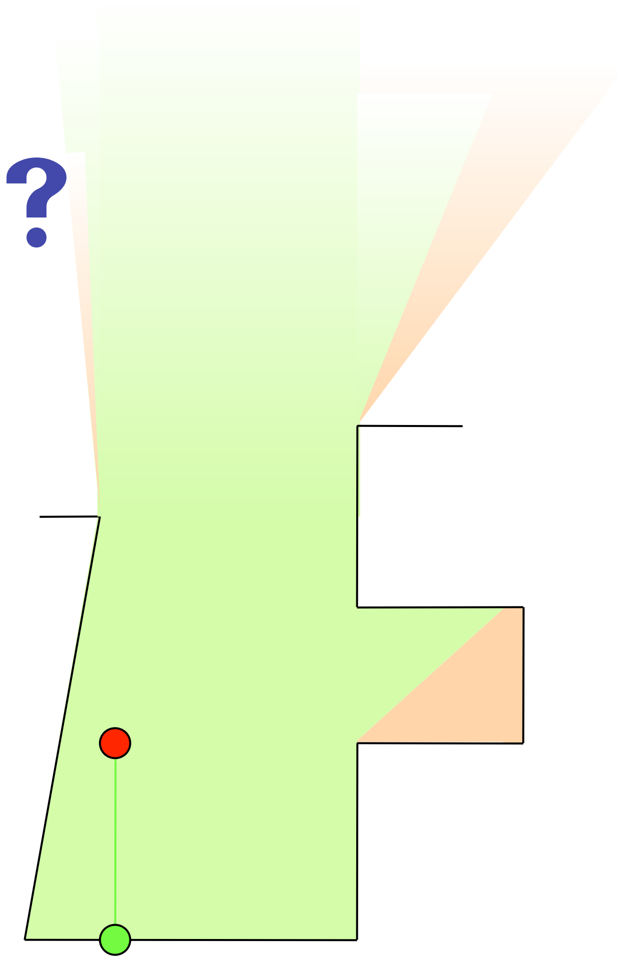


- Watchman problem
- Online, continuous vision:
  - optimum watchman route (L_1-metric) in simple rectilinear polygons (Deng et al.)
  - c=26.5 in simple polygons (Hoffmann et al.)
  - No competitive online algorithm for polygons with holes (Albers et al.)

Motivation

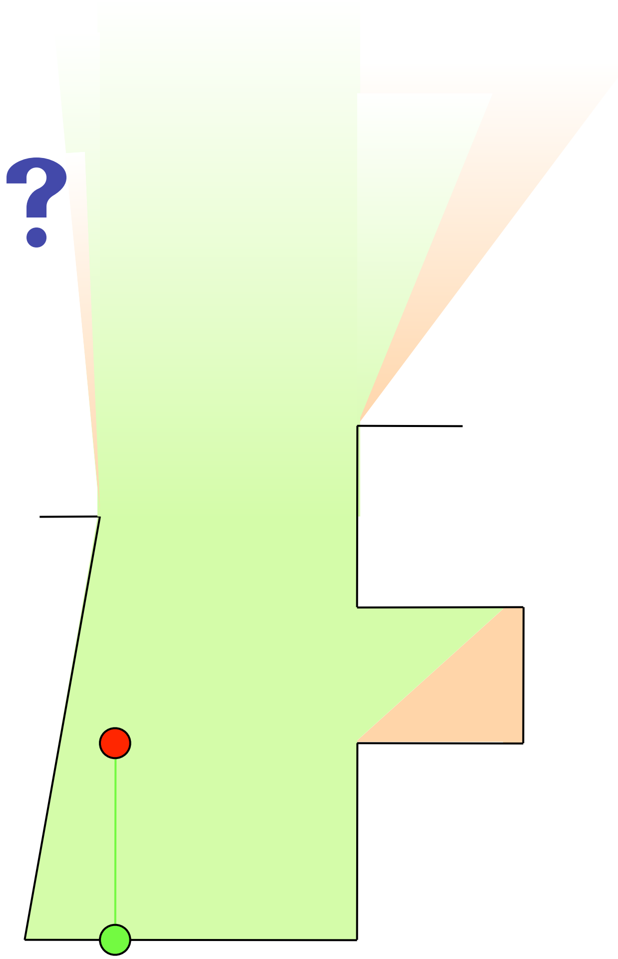


Motivation



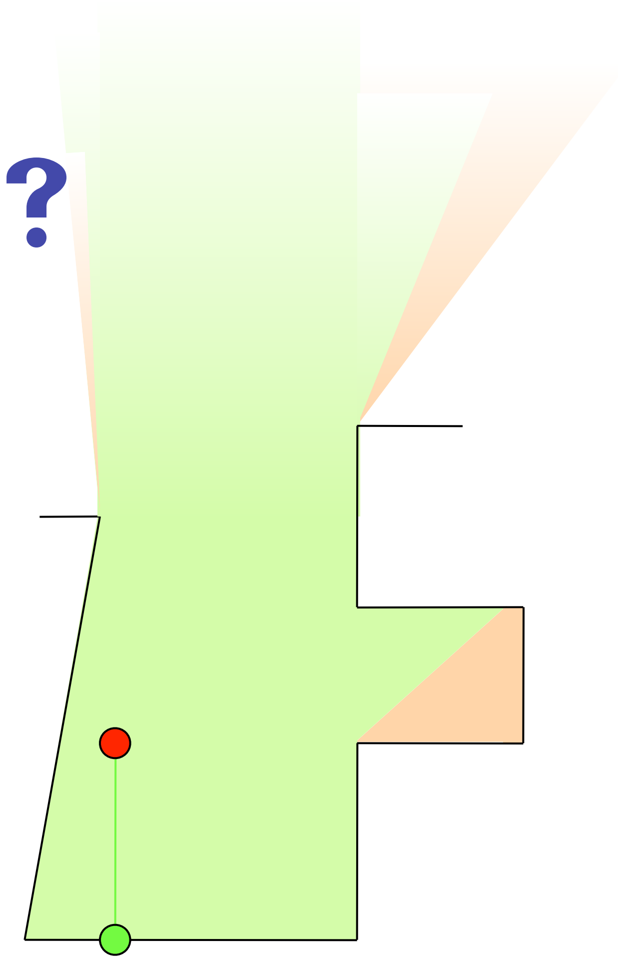
- Autonomous robot without continuous vision (scan costs)
- Watchman route

Motivation

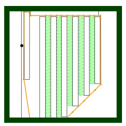


- Autonomous robot without continuous vision (scan costs)
- Watchman route
- Online problem

Motivation



- Autonomous robot without continuous vision (scan costs)
- Watchman route
- Online problem
- Several classes of polygons
- Is it possible to achieve a competitive strategy?



Polygons with Holes

Polygons with holes

Proposition:

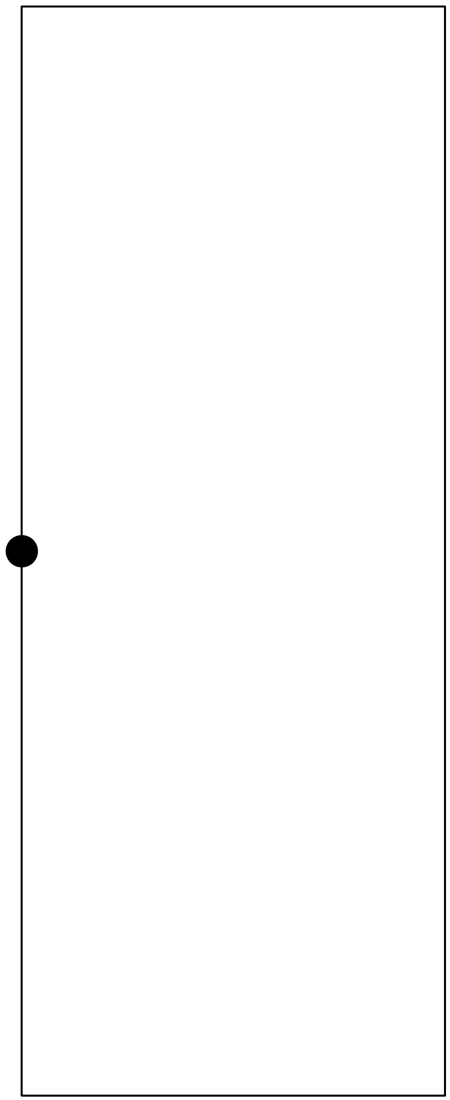
There is no strategy that achieves a bounded competitive ratio for the watchman problem with scan costs in case of a polygon with holes/obstacles.
This statement holds even if the polygon is rectilinear.

Proof of the proposition

Proof of the proposition

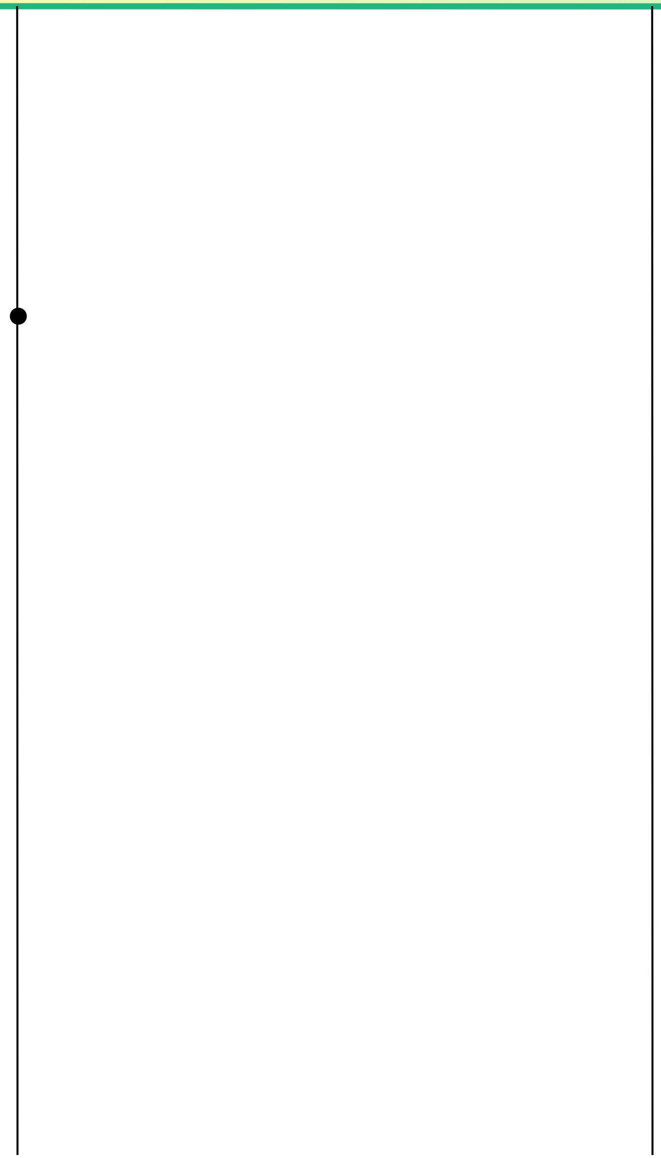
- Show: competitive ratio $\Omega(\sqrt{n})$
- Polygon with obstacles (panpipe)
- Further obstacles: placed depending on the strategy of the robot

Proof of the proposition



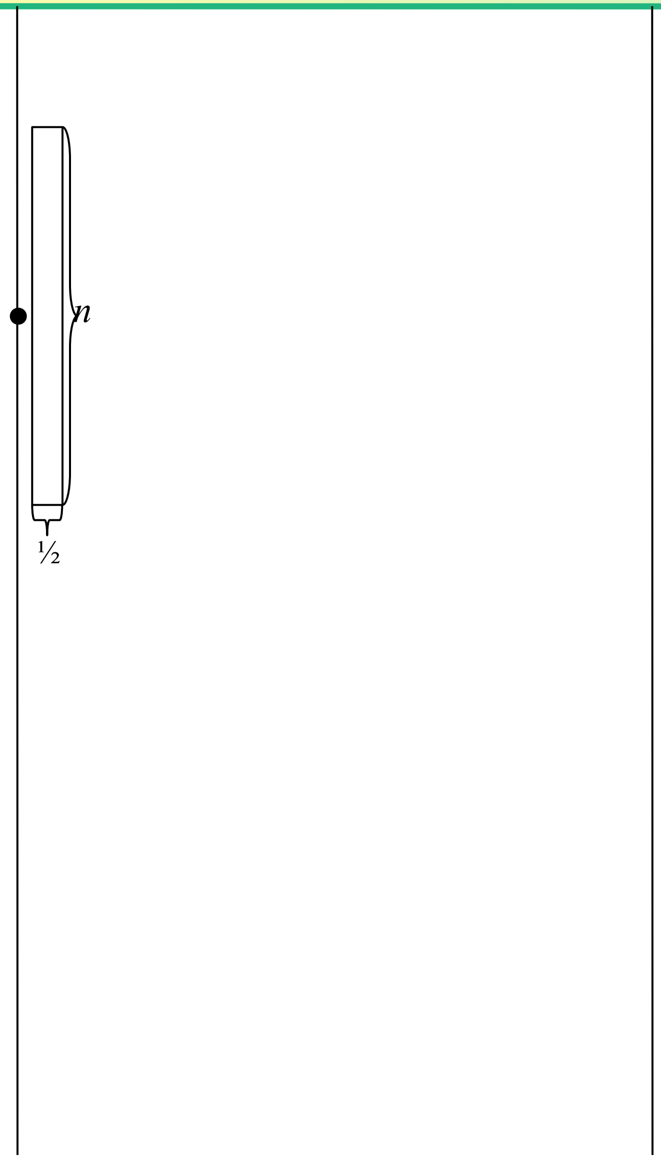
- Show: competitive ratio $\Omega(\sqrt{n})$
- Polygon with obstacles (panpipe)
- Further obstacles: placed depending on the strategy of the robot

Proof of the proposition



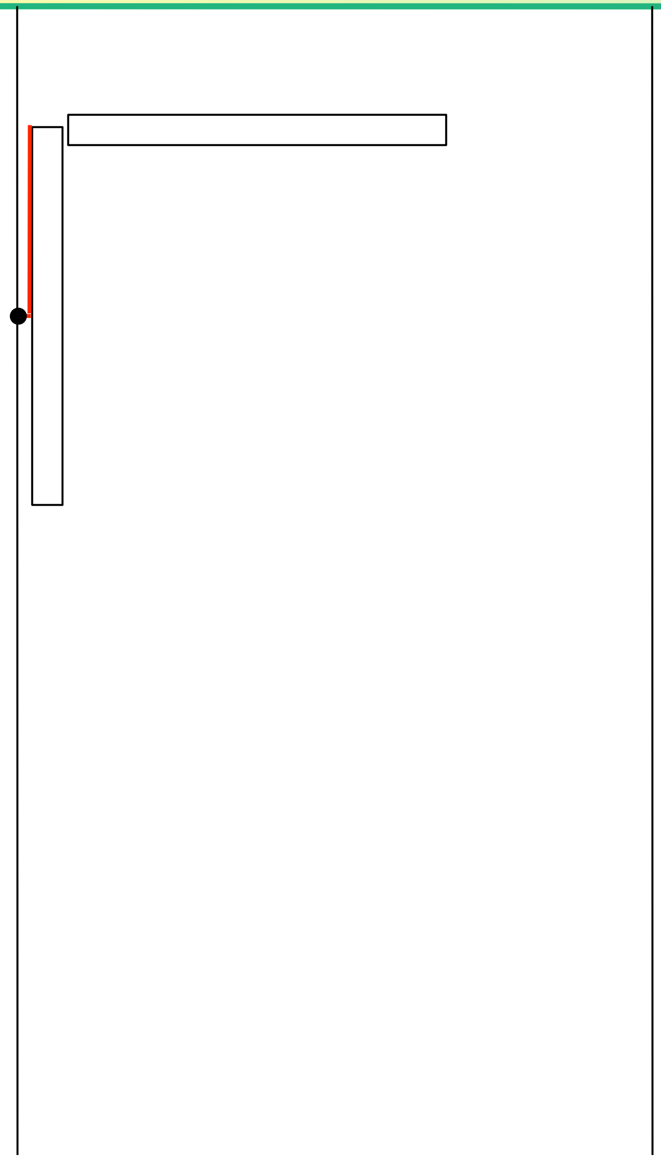
- Show: competitive ratio $\Omega(\sqrt{n})$
- Polygon with obstacles (panpipe)
- Further obstacles: placed depending on the strategy of the robot

Proof of the proposition



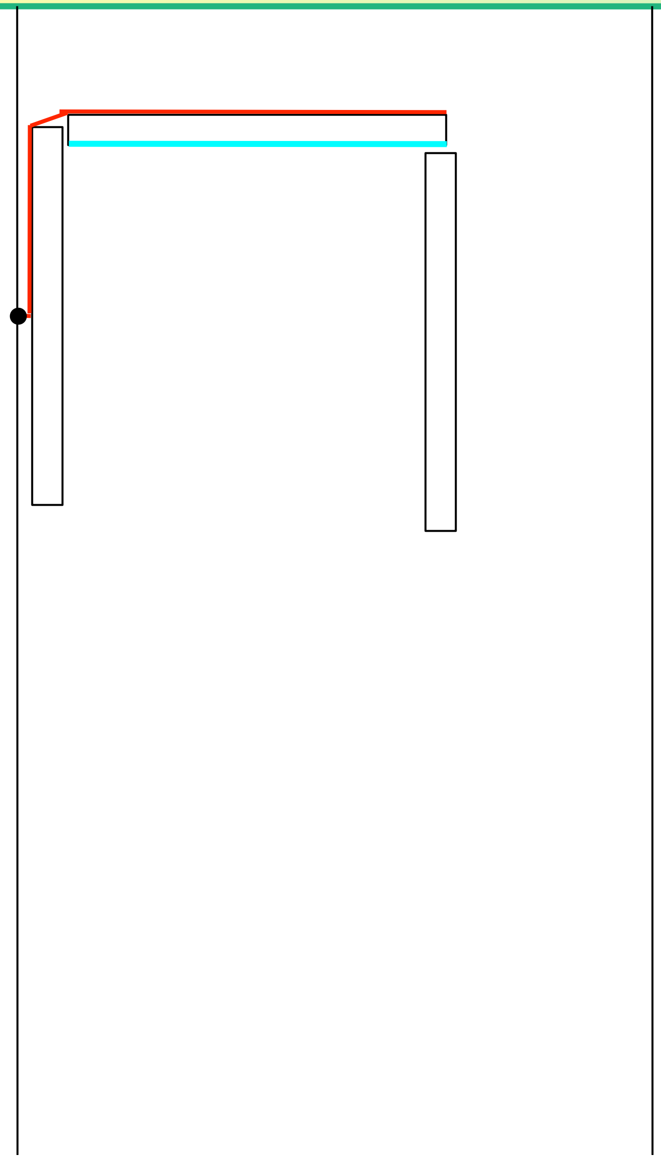
- Show: competitive ratio $\Omega(\sqrt{n})$
- Polygon with obstacles (panpipe)
- Further obstacles: placed depending on the strategy of the robot

Proof of the proposition



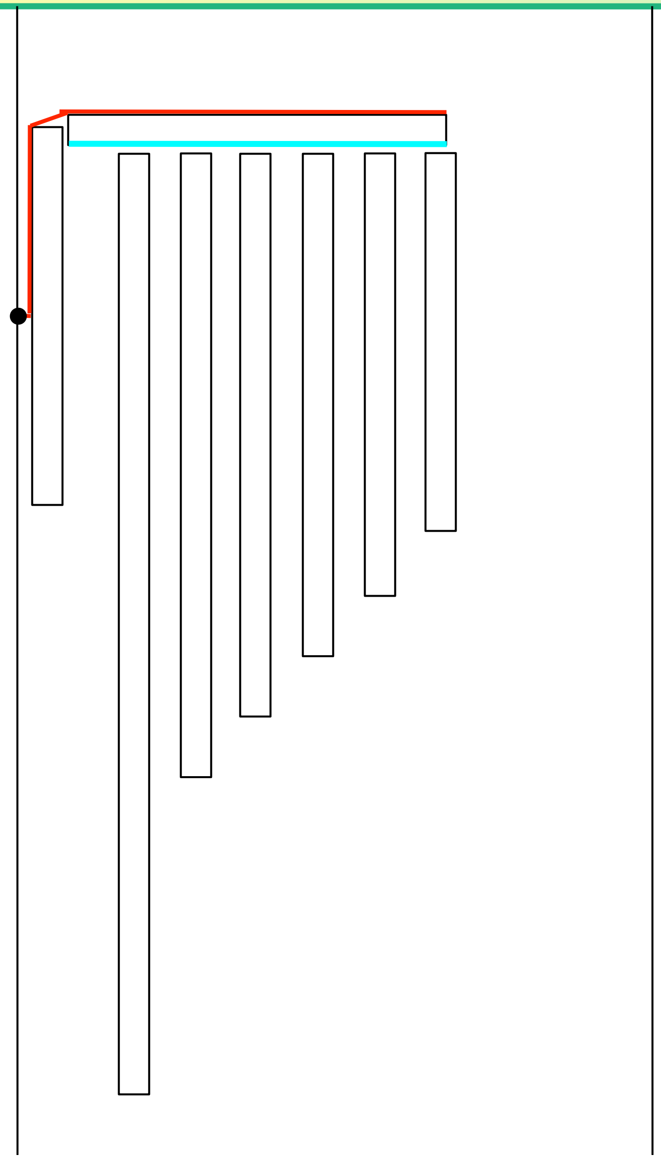
- Show: competitive ratio $\Omega(\sqrt{n})$
- Polygon with obstacles (panpipe)
- Further obstacles: placed depending on the strategy of the robot

Proof of the proposition



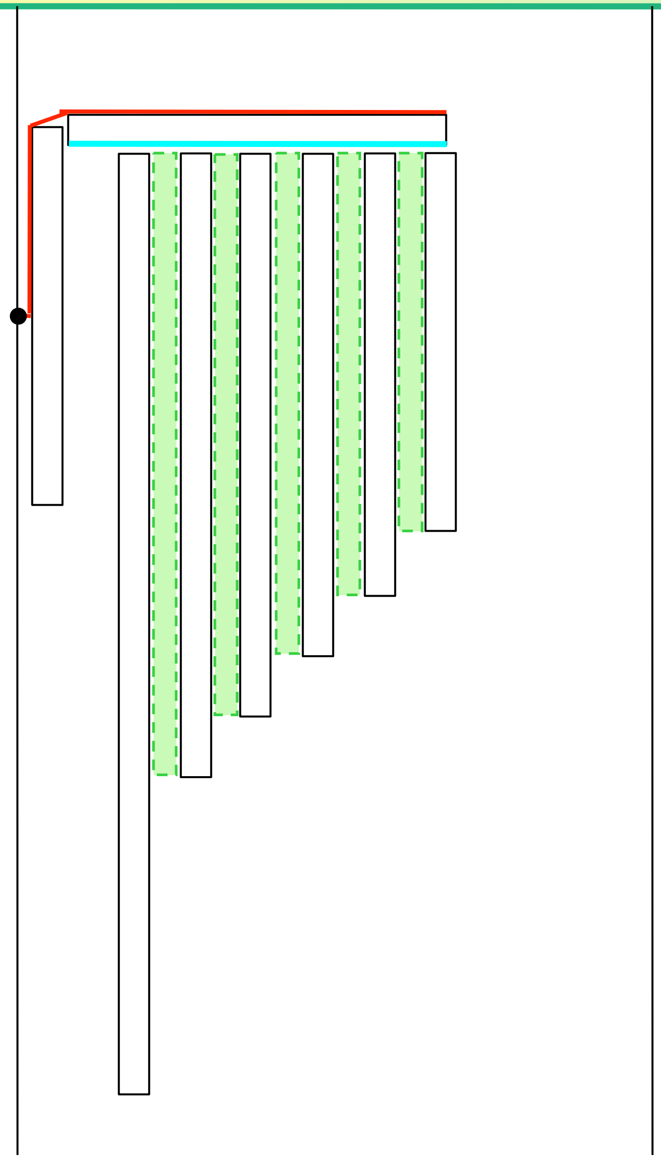
- Show: competitive ratio $\Omega(\sqrt{n})$
- Polygon with obstacles (panpipe)
- Further obstacles: placed depending on the strategy of the robot

Proof of the proposition



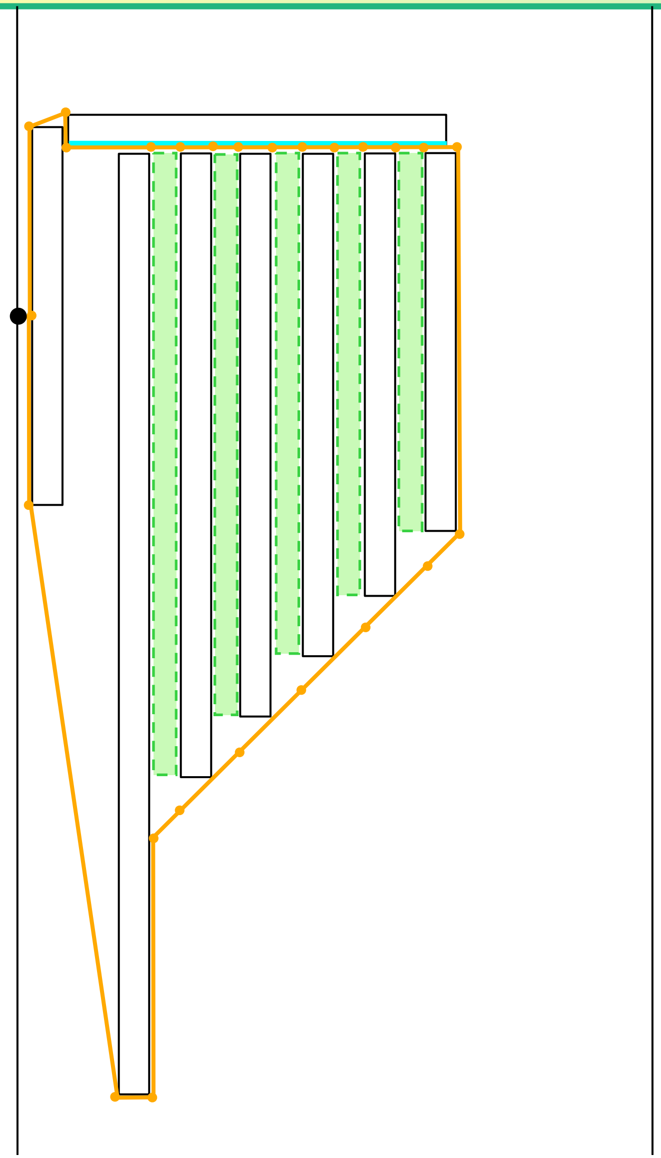
- Show: competitive ratio $\Omega(\sqrt{n})$
- Polygon with obstacles (panpipe)
- Further obstacles: placed depending on the strategy of the robot

Proof of the proposition



- Show: competitive ratio $\Omega(\sqrt{n})$
- Polygon with obstacles (panpipe)
- Further obstacles: placed depending on the strategy of the robot

Proof of the proposition



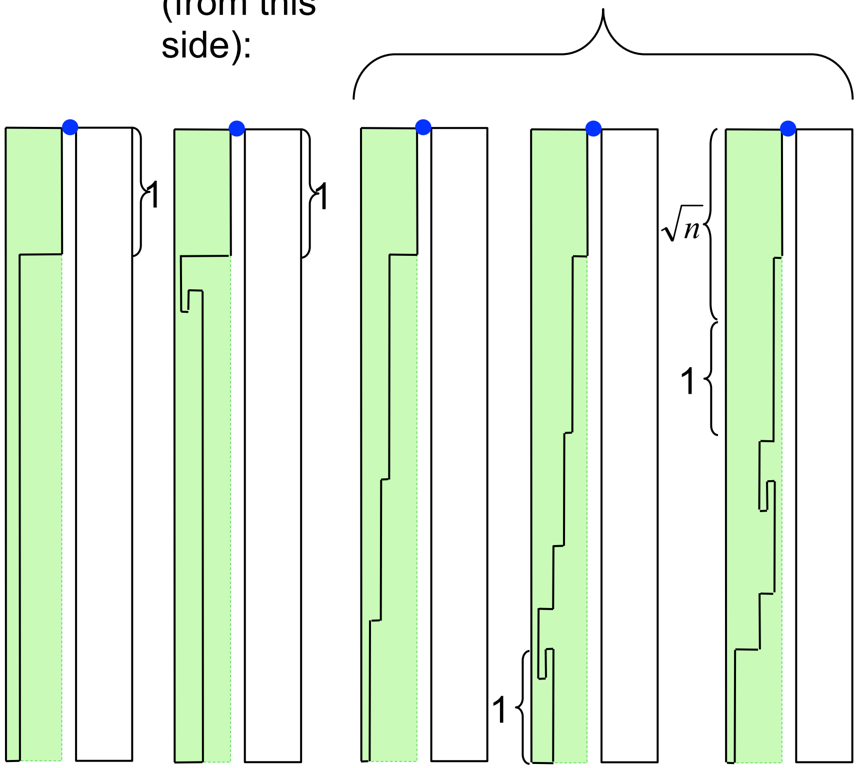
- Show: competitive ratio $\Omega(\sqrt{n})$
- Polygon with obstacles (panpipe)
- Further obstacles: placed depending on the strategy of the robot

Proof of the proposition

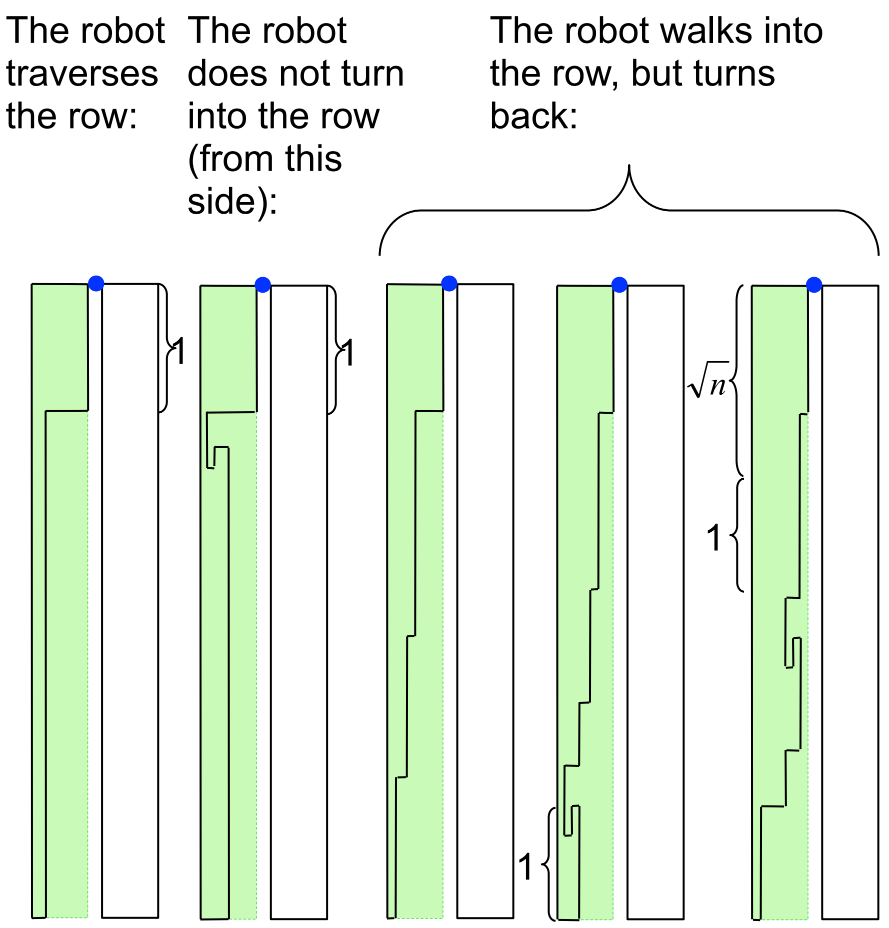
The robot traverses the row:

The robot does not turn into the row (from this side):

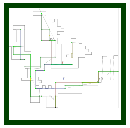
The robot walks into the row, but turns back:



Proof of the proposition

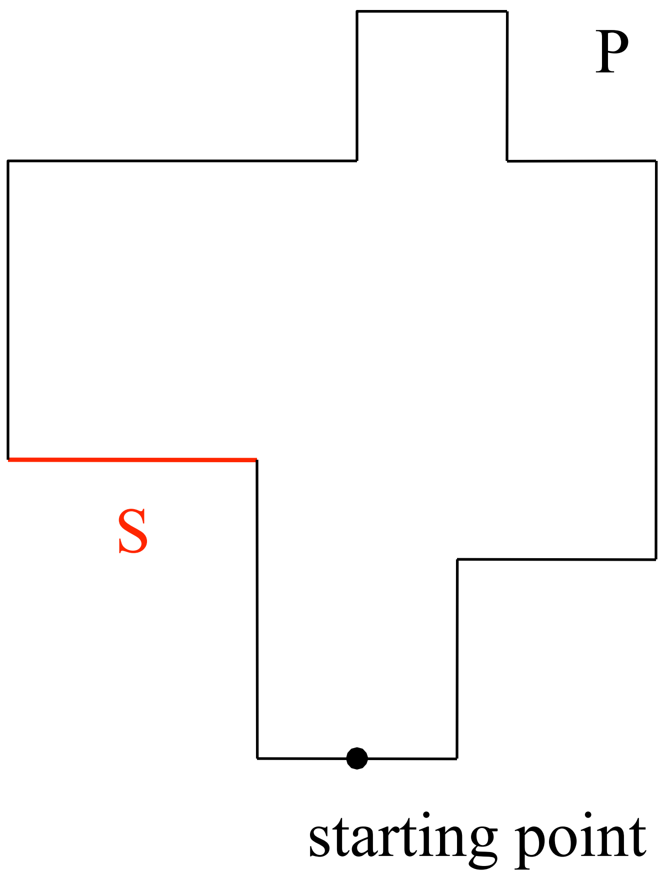


- The shape of the inserted objects depends on the path of the robot.

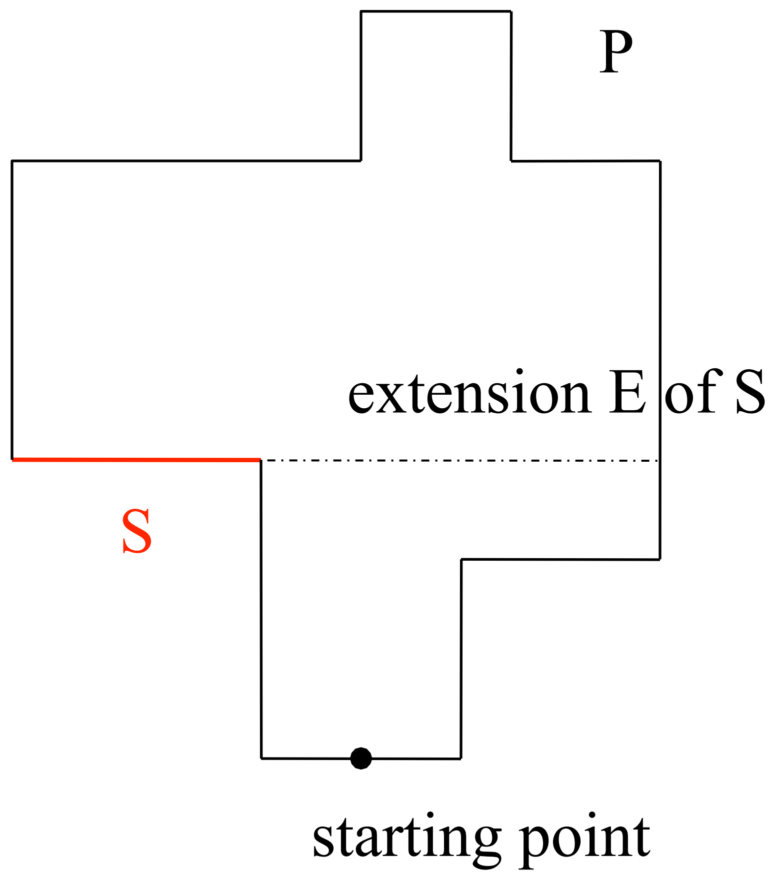


A Competitive Strategy for Simple Rectilinear Polygons

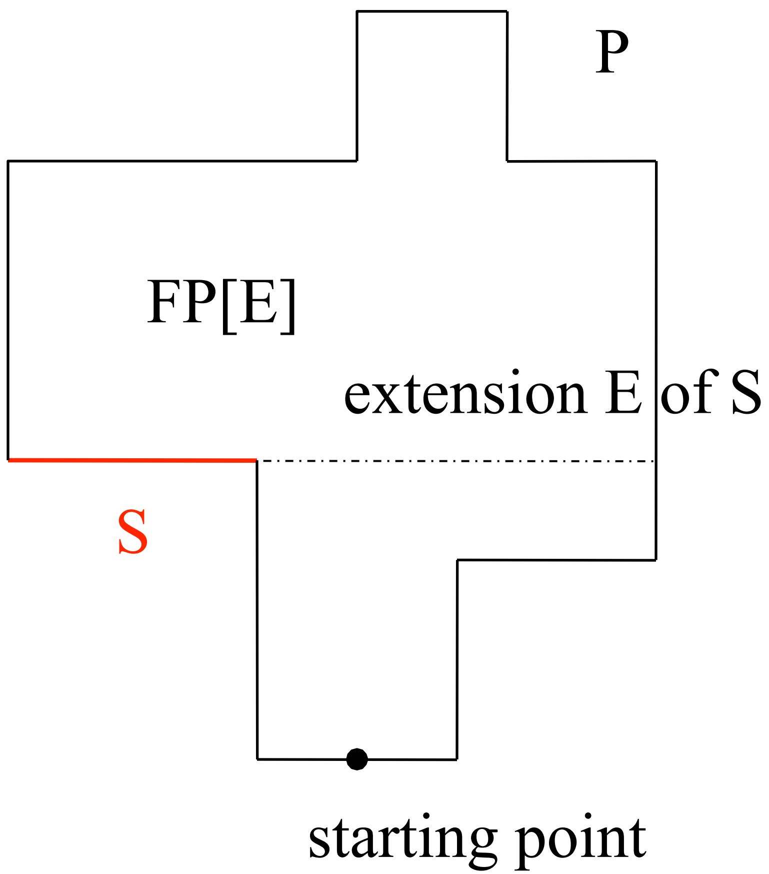
Extensions



Extensions

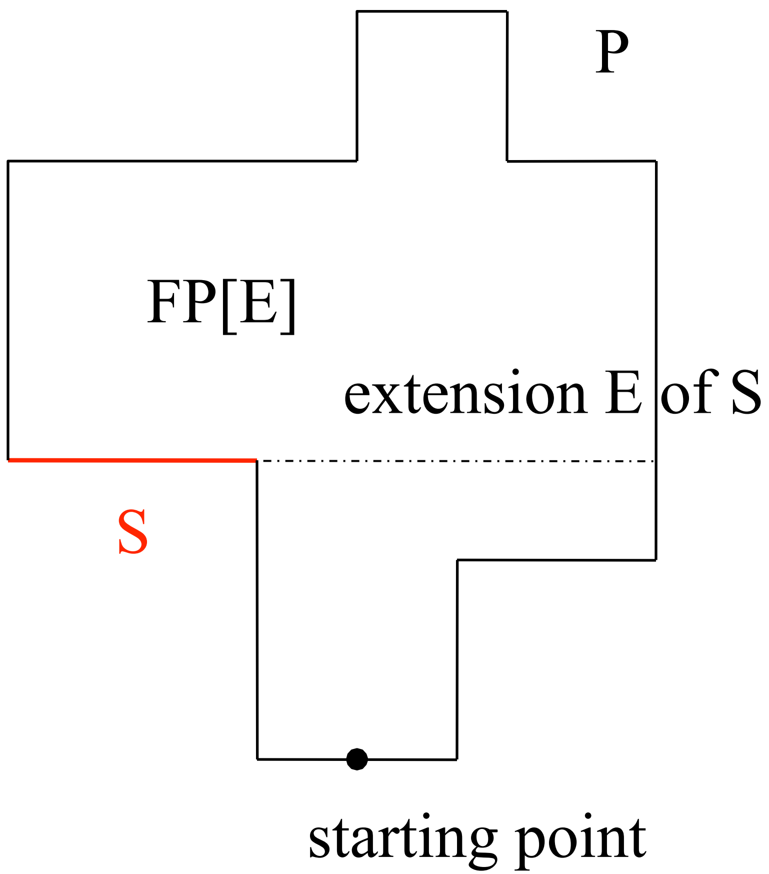


Extensions



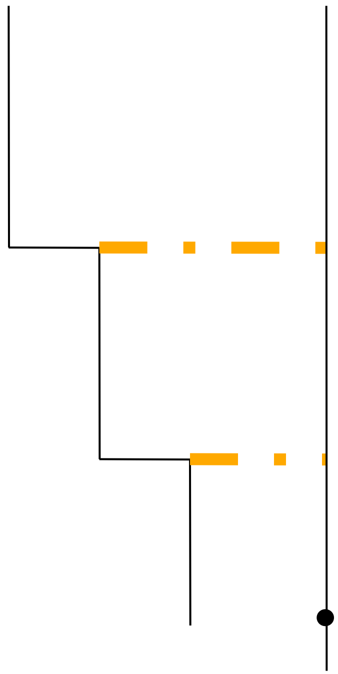
- Two subpolygons
- Necessary and essential extensions

Extensions



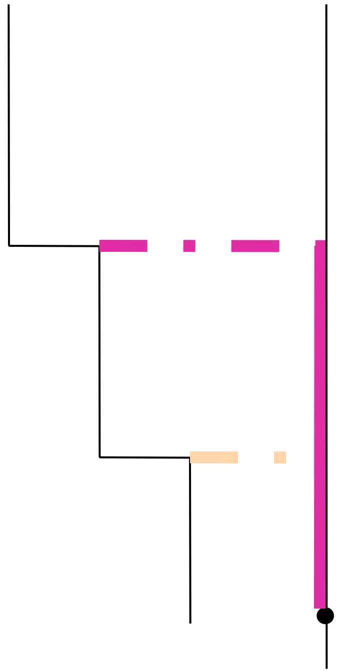
- Two subpolygons
- Necessary and essential extensions
- Advantage in rectilinear polygons

Extensions



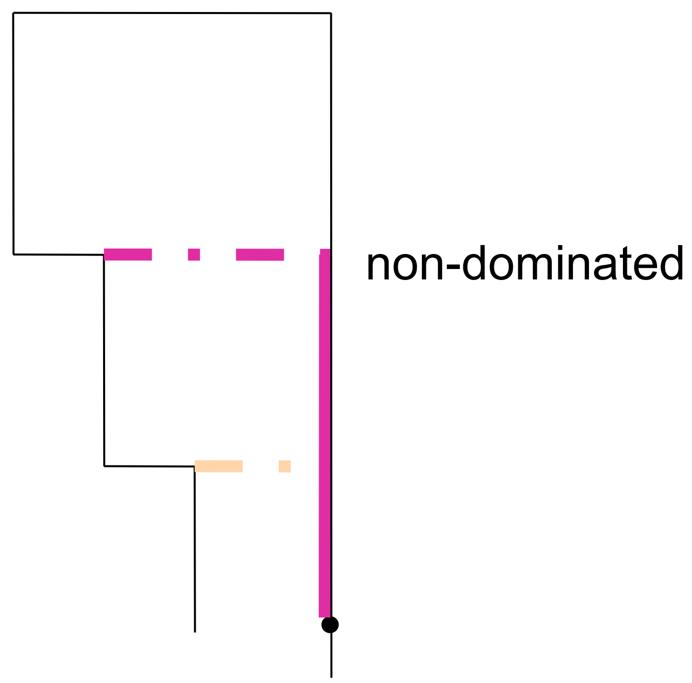
- Two subpolygons
- Necessary and essential extensions
- Advantage in rectilinear polygons

Extensions



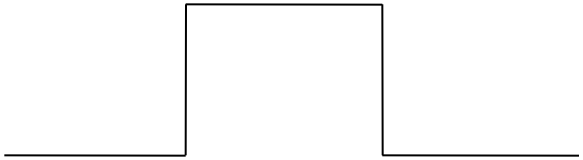
- Two subpolygons
- Necessary and essential extensions
- Advantage in rectilinear polygons

Extensions

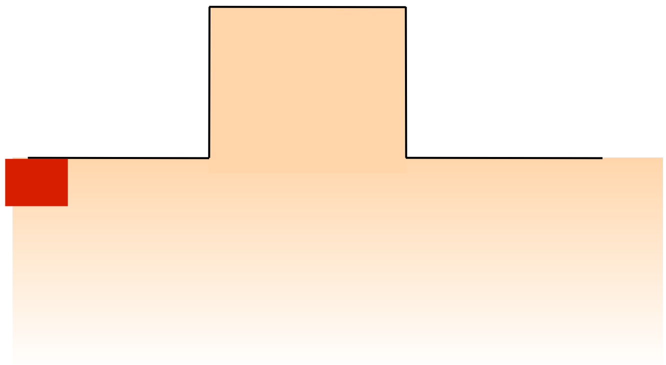


- Two subpolygons
- Necessary and essential extensions
- Advantage in rectilinear polygons

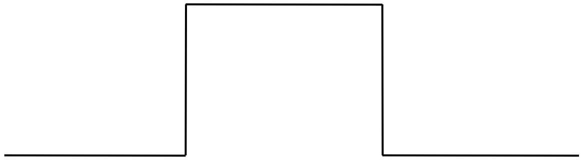
A competitive strategy for simple rectilinear polygons



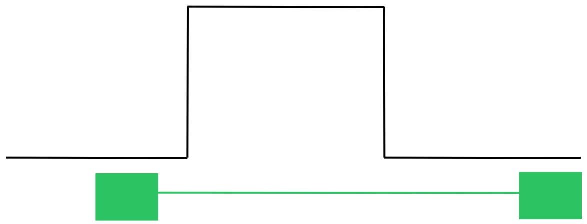
A competitive strategy for simple rectilinear polygons



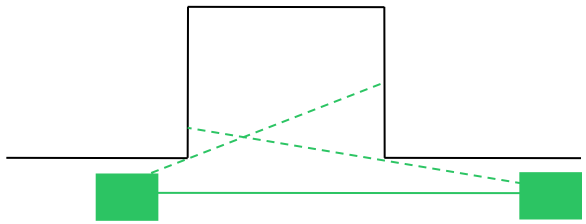
A competitive strategy for simple rectilinear polygons



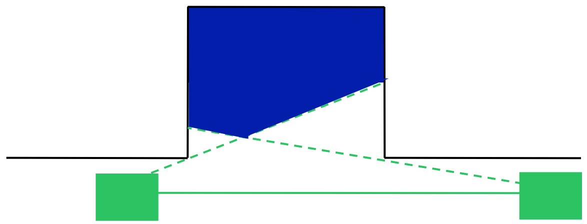
A competitive strategy for simple rectilinear polygons



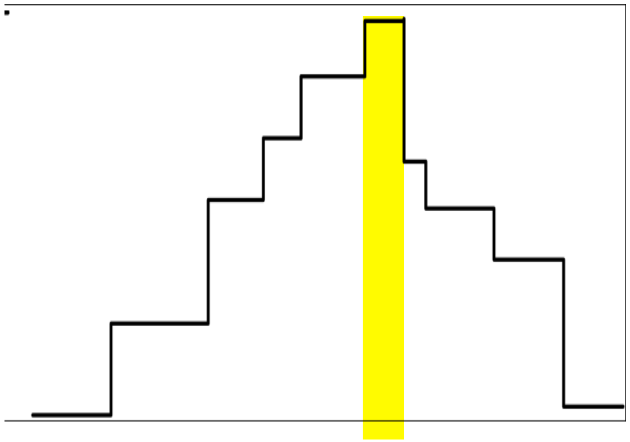
A competitive strategy for simple rectilinear polygons



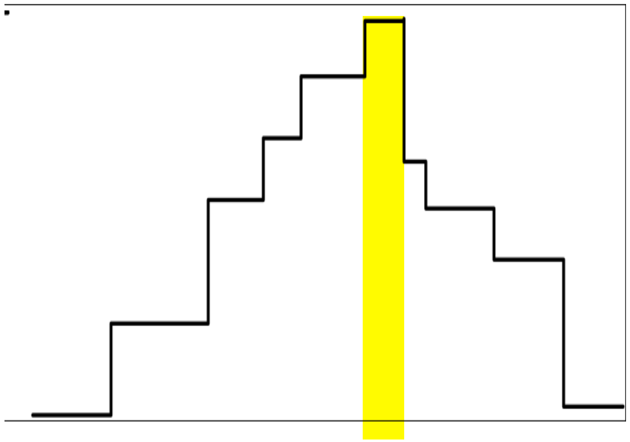
A competitive strategy for simple rectilinear polygons



A competitive strategy for simple rectilinear polygons



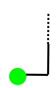
A competitive strategy for simple rectilinear polygons



- Problem with niches
- It is necessary to limit the number of scan points

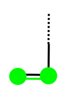
A competitive strategy for simple rectilinear polygons

A competitive strategy for simple rectilinear polygons



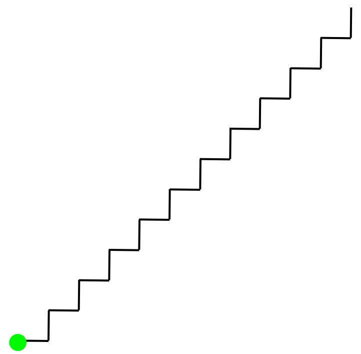
- Minimum side length a
- Consider the distance to the next corner (reflex vertex): walk beyond the corner if the distance to it is “short”

A competitive strategy for simple rectilinear polygons



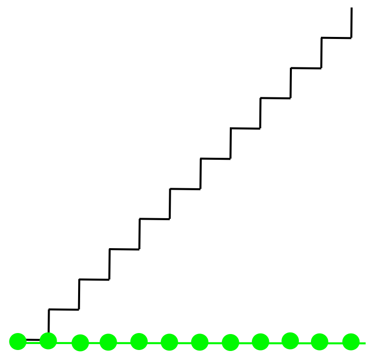
- Minimum side length a
- Consider the distance to the next corner (reflex vertex): walk beyond the corner if the distance to it is "short"

A competitive strategy for simple rectilinear polygons



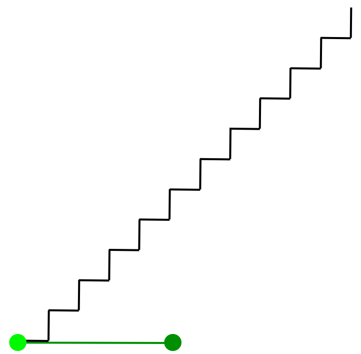
- Minimum side length a
- Consider the distance to the next corner (reflex vertex): walk beyond the corner if the distance to it is “short”

A competitive strategy for simple rectilinear polygons



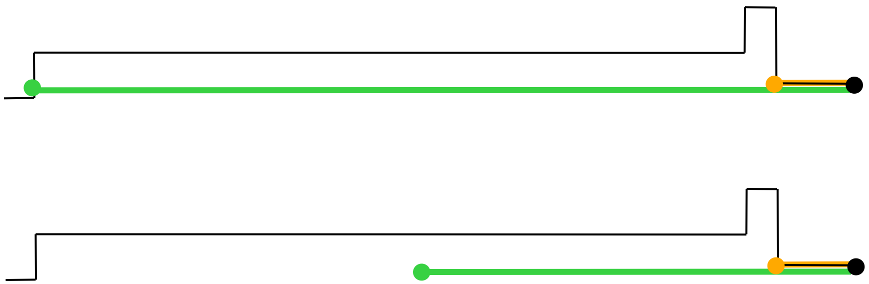
- Minimum side length a
- Consider the distance to the next corner (reflex vertex): walk beyond the corner if the distance to it is "short"

A competitive strategy for simple rectilinear polygons



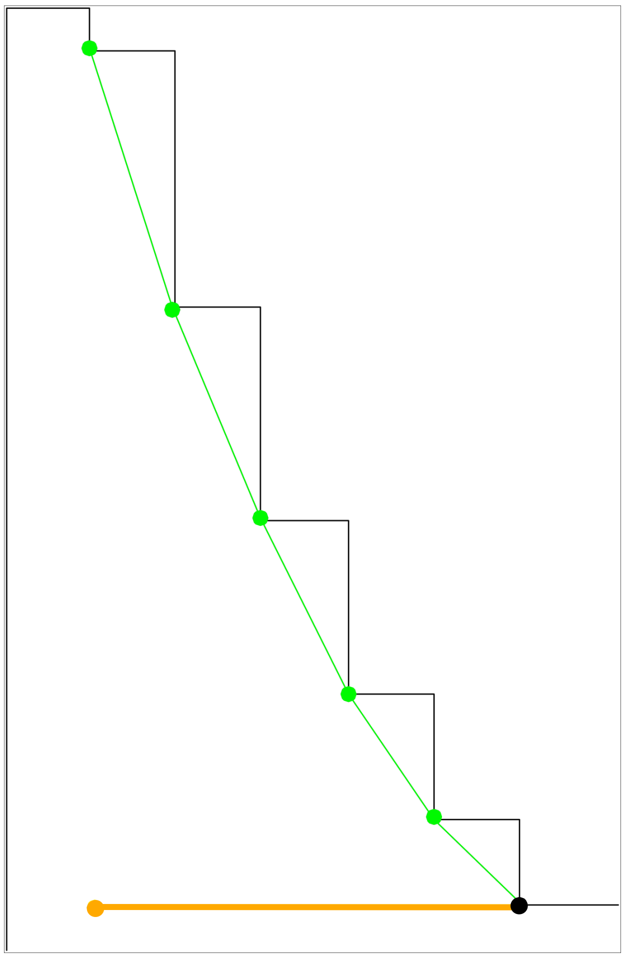
- Minimum side length a
- Consider the distance to the next corner (reflex vertex): walk beyond the corner if the distance to it is "short"

A competitive strategy for simple rectilinear polygons



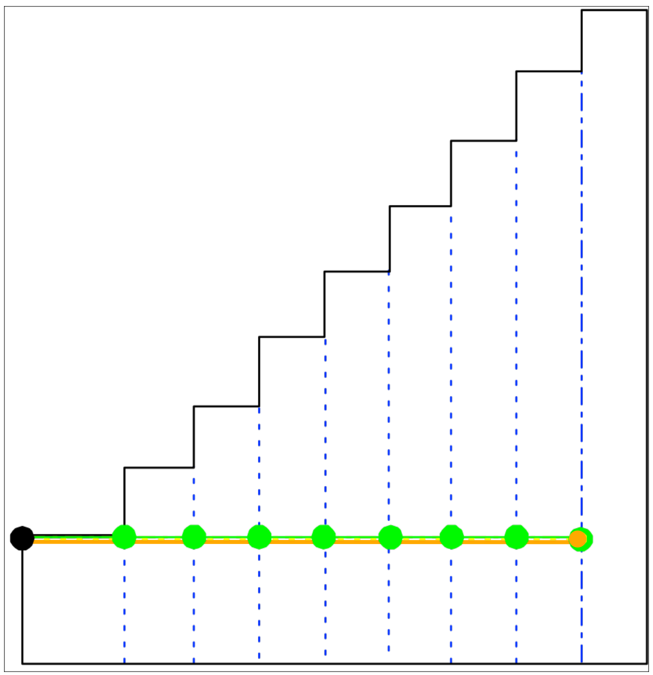
- Minimum side length a
- Consider the distance to the next corner (reflex vertex): walk beyond the corner if the distance to it is "short"
- Adapt the step length of the robot to the minimal necessary step length

A competitive strategy for simple rectilinear polygons



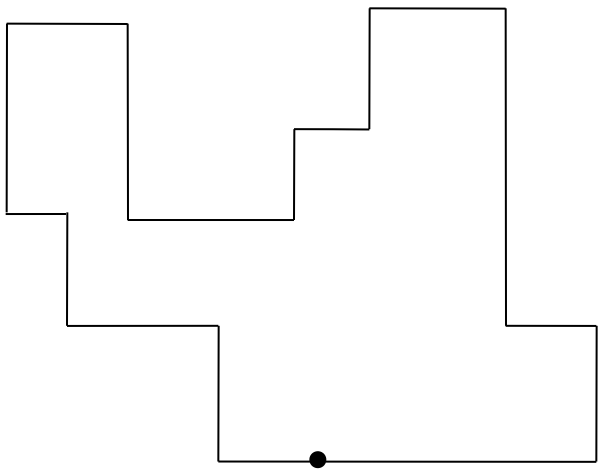
- Minimum side length a
- Consider the distance to the next corner (reflex vertex): walk beyond the corner if the distance to it is “short”
- Adapt the step length of the robot to the minimal necessary step length
- Move to the projection of a corner and not to the corner itself

A competitive strategy for simple rectilinear polygons

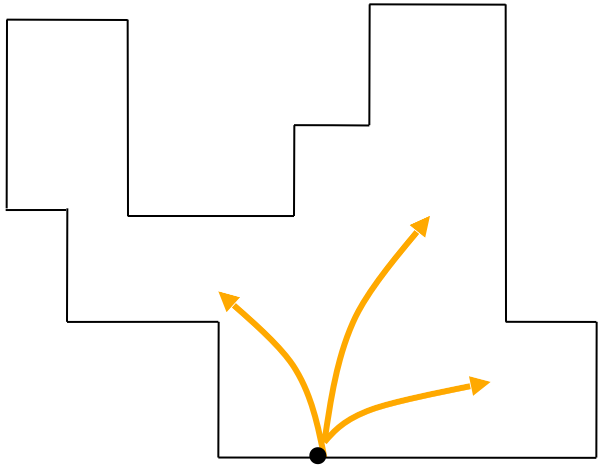


- Minimum side length a
- Consider the distance to the next corner (reflex vertex): walk beyond the corner if the distance to it is “short”
- Adapt the step length of the robot to the minimal necessary step length
- Move to the projection of a corner and not to the corner itself
- Do not scan on each necessary extension

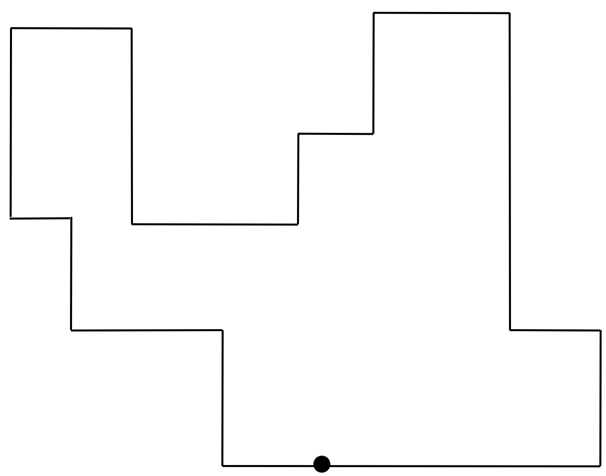
Order of extensions



Order of extensions

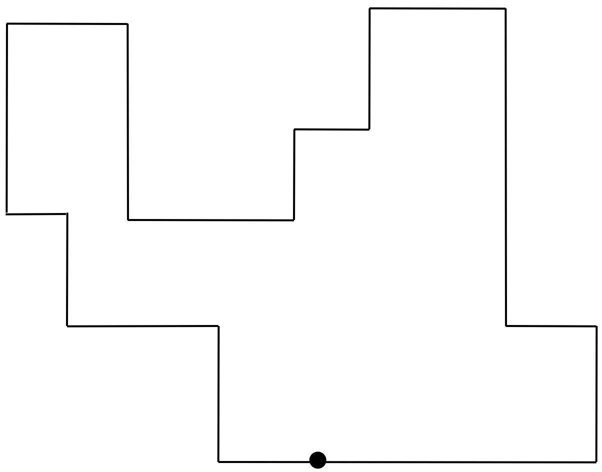


Order of extensions



- GREEDY-ONLINE algorithm for a robot with continuous vision.
- Based on a proposition of Chin and Ntafos:

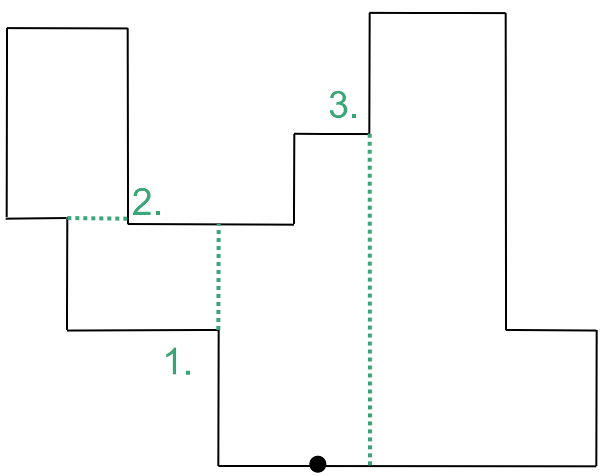
Order of extensions



- GREEDY-ONLINE algorithm for a robot with continuous vision.
- Based on a proposition of Chin and Ntafos:

  Any optimum watchman route in P, a simple rectilinear polygon, will have to visit the essential edges in the order in which they appear on the boundary of P' (the new polygon obtained by removing the "non-essential" portions of the polygon).
- Transfer of this proposition.

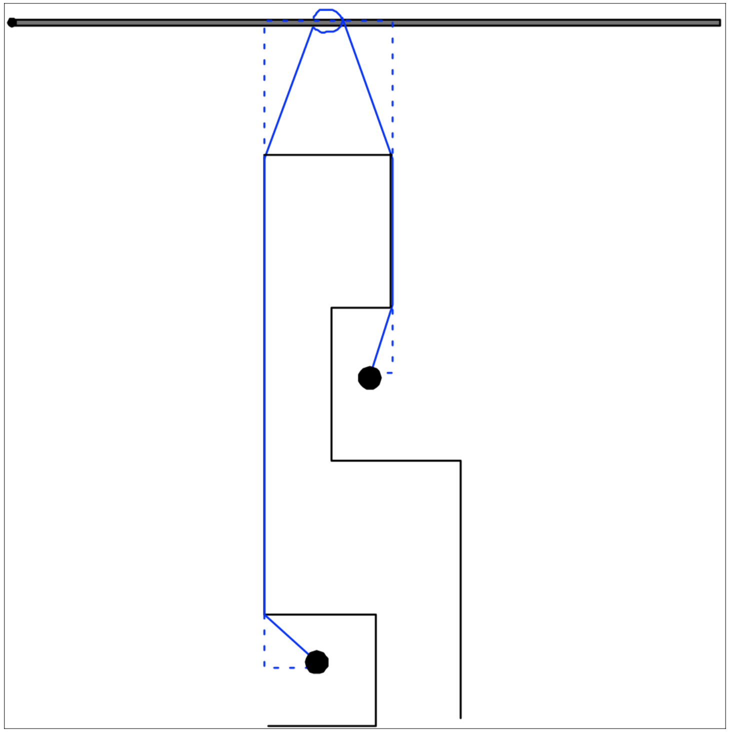
Order of extensions



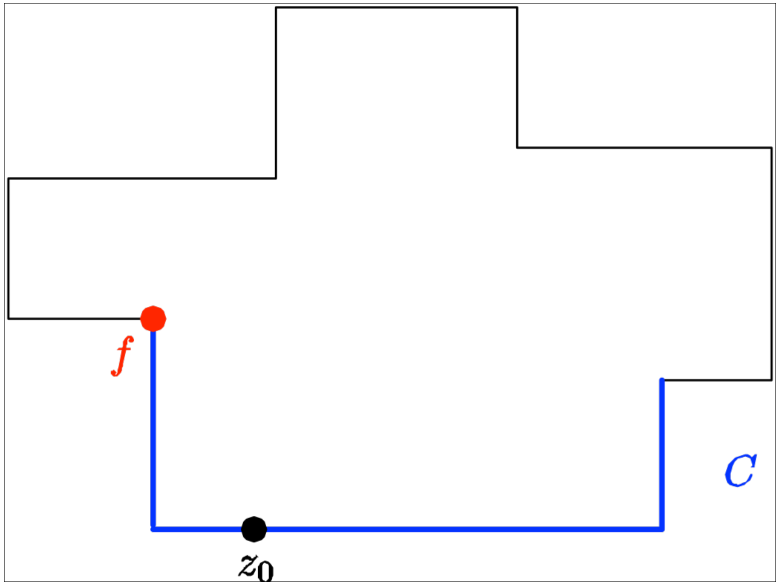
- GREEDY-ONLINE algorithm for a robot with continuous vision.
- Based on a proposition of Chin and Ntafos:

  Any optimum watchman route in P, a simple rectilinear polygon, will have to visit the essential edges in the order in which they appear on the boundary of P' (the new polygon obtained by removing the "non-essential" portions of the polygon).
- Transfer of this proposition.

GREEDY-ONLINE algorithm

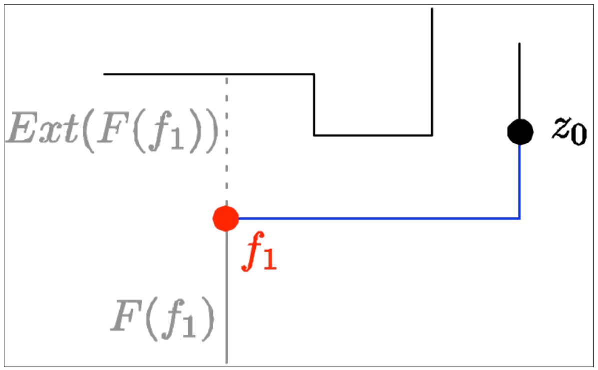


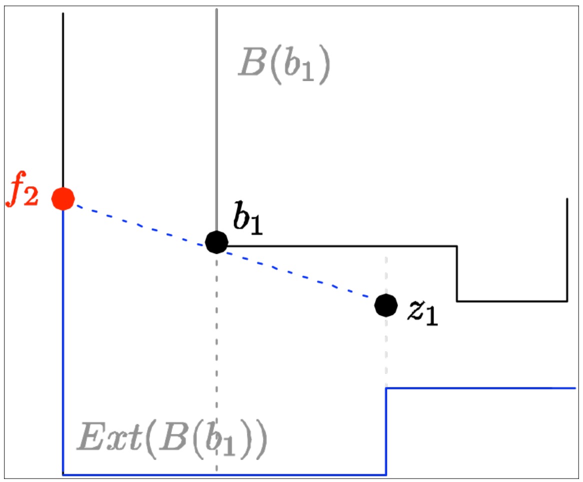
GREEDY-ONLINE algorithm



- Taut-Thread-Principle
- Consider the contiguous part of the boundary that was already visible from some point of the route
- Either f is a 270° corner or a corner blocks the sight such as only f^- is visible

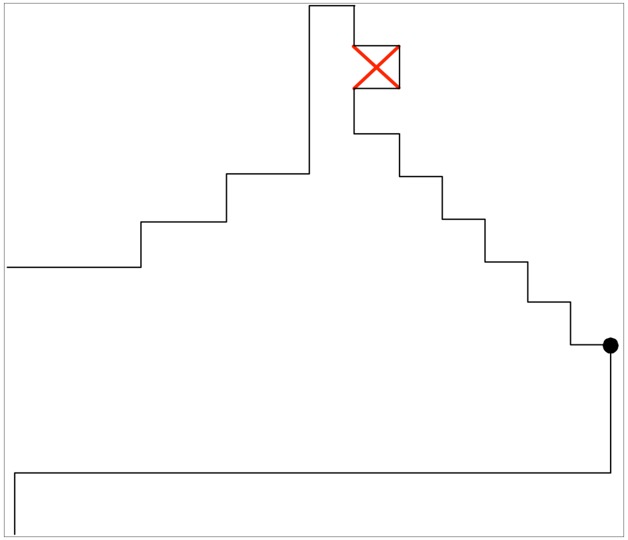
GREEDY-ONLINE algorithm





- Taut-Thread-Principle
- Consider the contiguous part of the boundary that was already visible from some point of the route
- Either f is a 270° corner or a corner blocks the sight such as only f^- is visible

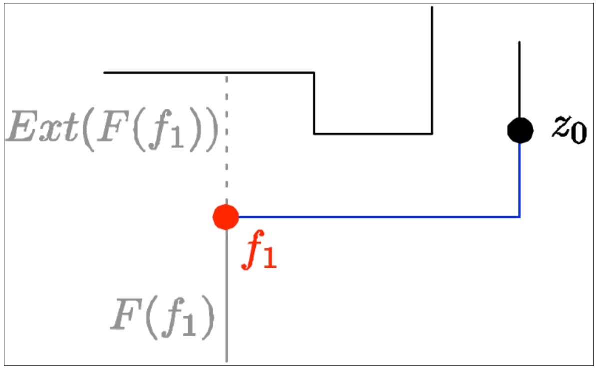
GREEDY-ONLINE algorithm

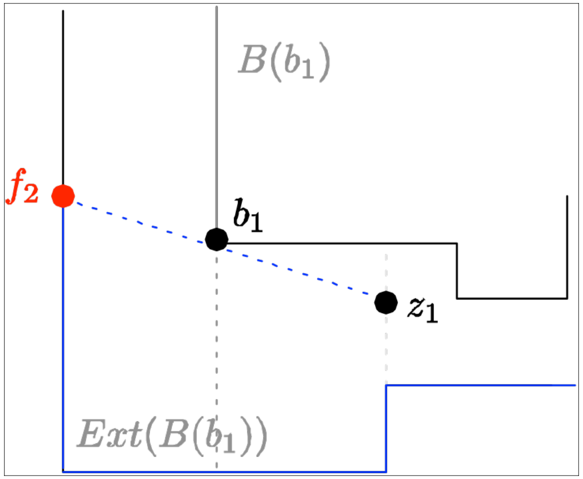


- Taut-Thread-Principle
- Consider the contiguous part of the boundary that was already visible from some point of the route
- Either f is a 270° corner or a corner blocks the sight such as only f^- is visible

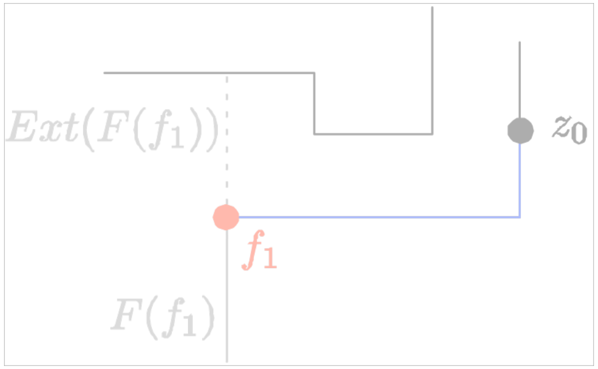
A competitive strategy for simple rectilinear polygons

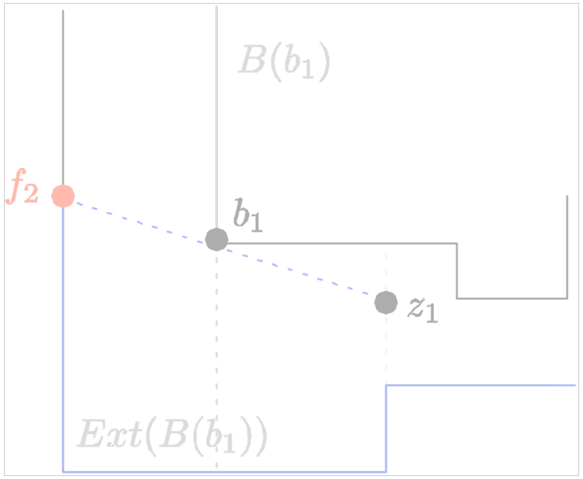
A competitive strategy for simple rectilinear polygons





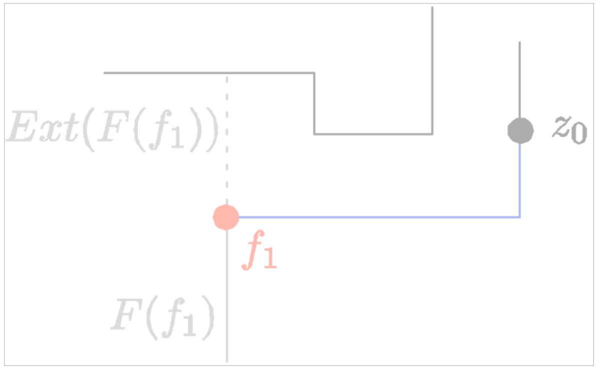
A competitive strategy for simple rectilinear polygons

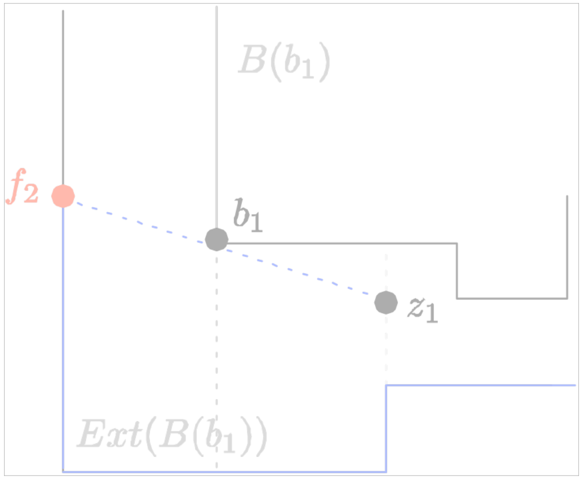




- Extensions of the GREEDY-ONLINE algorithm
- Interval case vs. extension case

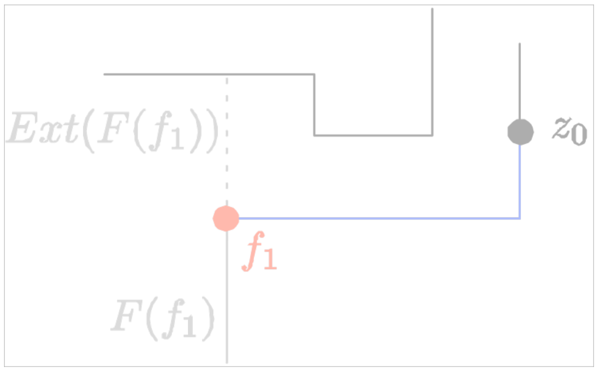
A competitive strategy for simple rectilinear polygons

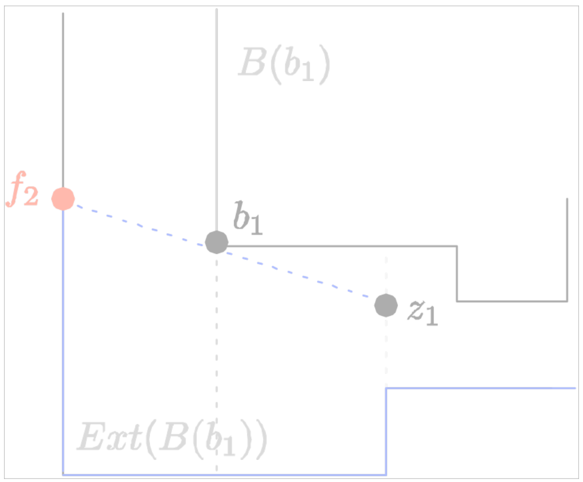




- Extensions of the GREEDY-ONLINE algorithm
- Interval case vs. extension case
- Reaching the extension on an axis-parallel path without a change of direction is possible/impossible

A competitive strategy for simple rectilinear polygons



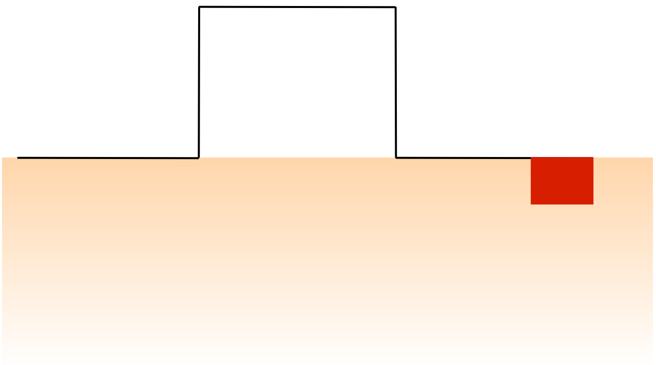


- Extensions of the GREEDY-ONLINE algorithm
- Interval case vs. extension case
- Reaching the extension on an axis-parallel path without a change of direction is possible/impossible
- In all cases of the case differentiation:
  - In case the robot runs beyond the extension: the robot is (is not) able to cover the total planned length
  - Positive line creation vs. negative line creation

Binary search in the strategy

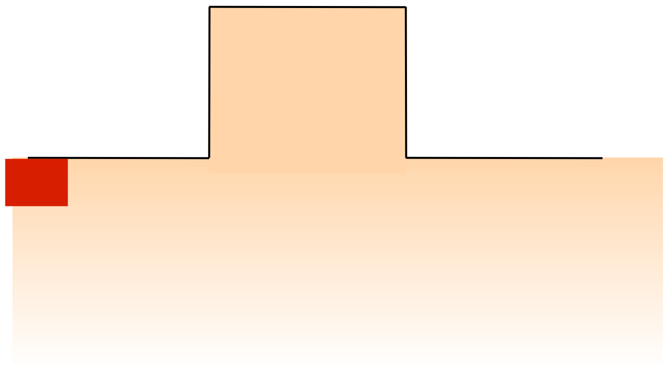
- Non-visible region (NVR): An area in which the parts of the boundary, which would be visible by simply passing them with continuous vision, are not yet completely visible.
- Discover passed non-visible regions with binary search.

Binary search in the strategy



- Non-visible region (NVR): An area in which the parts of the boundary, which would be visible by simply passing them with continuous vision, are not yet completely visible.
- Discover passed non-visible regions with binary search.

Binary search in the strategy

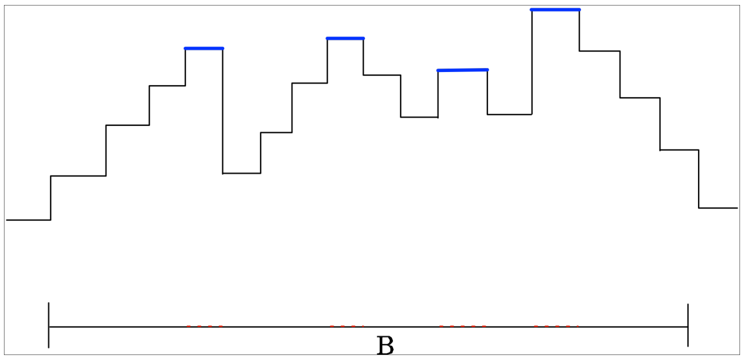


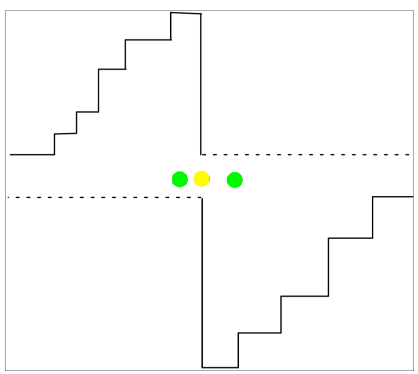
- Non-visible region (NVR): An area in which the parts of the boundary, which would be visible by simply passing them with continuous vision, are not yet completely visible.
- Discover passed non-visible regions with binary search.

Binary search in the strategy

- Non-visible region (NVR): An area in which the parts of the boundary, which would be visible by simply passing them with continuous vision, are not yet completely visible.
- Discover passed non-visible regions with binary search.

Binary search in the strategy





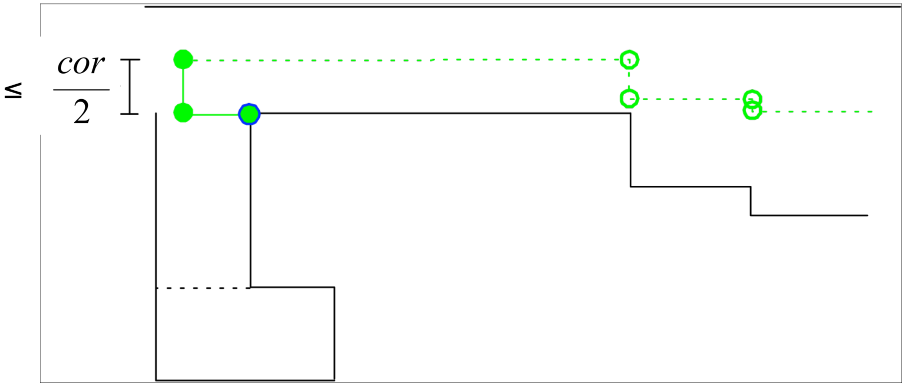
- If the optimum nees k scans in an interval, the robot which uses the strategy will need maximum
  - k binary searches (for each an upper bound is given) or
  - 2k binary searches if the NVRs may appear on two sides

Turn adjustments

Turn adjustments

- The optimum may have the opportunity to turn off before the robot, following the strategy, does.
- The robot may discover a corridor inside a non-visible region.
- Ø Adjustments to have the best basic position for the next turn

Turn adjustments



- The optimum may have the opportunity to turn off before the robot, following the strategy, does.
- The robot may discover a corridor inside a non-visible region.
- Adjustments to have the best basic position for the next turn
- Minimum corridor width a_k

The strategy

$a \leq 1$:

A. An axis-parallel move to E is possible without a turn

- e≥2a+1: interval case

  Let d_i be the actual distance to the perpendicular of the next counterclockwise extension

  – If $d_i > 2a+1$, move to the perpendicular of the corner

  – If $d_i \leq 2a+1$: If $d_i > a$: cover a distance of $2d_i+1$

  If $d_i \leq a$: cover a distance of $2a+1$

  Apply binary search if necessary, that means, if non-visible regions appear.

  – If no corner appears on the counterclockwise side, move directly to E.

  In case we run beyond E with a step of length $2d_i+1/2a+1$:

  i. If we do not cover the total distance, because of the boundary: Run as far as possible, go back to E, move back in steps of length 1, apply binary search for NVRs (on the counterclockwise side till E, on both sides beyond E) and if a corridor is identified, use it and make turn adjustments

  ii. If we may cover the total distance:

  I. negative line creation: Apply binary search, if a corridor is discovered inside a NVR, use it and make turn adjustments.

  II. Positive line creation: Go back to E, move back in steps of length 1, apply binary search and search for a corridor and the critical extension, make turn adjustments.

- e<2a+1: extension case

  Cover a distance of 2e+1. In case:..(i., ii.)

The strategy

$a \leq 1$:

A. An axis-parallel move to E is possible without a turn
   - $e \geq 2a+1$: interval case
   - $e < 2a+1$: extension case

B. An axis-parallel move to E is not possible without a change of direction: Let b_i be the distance to the sight-blocking corner.
   - $e \geq a+1$: interval case
     - No non-visible region up to the sight-blocking corner
     - Along the boundary up to the sight-blocking corner occur non-visible regions
   - $e < a+1$: extension case

$a > 1$:

Similar; with scans every time a distance of a is covered.

The strategy

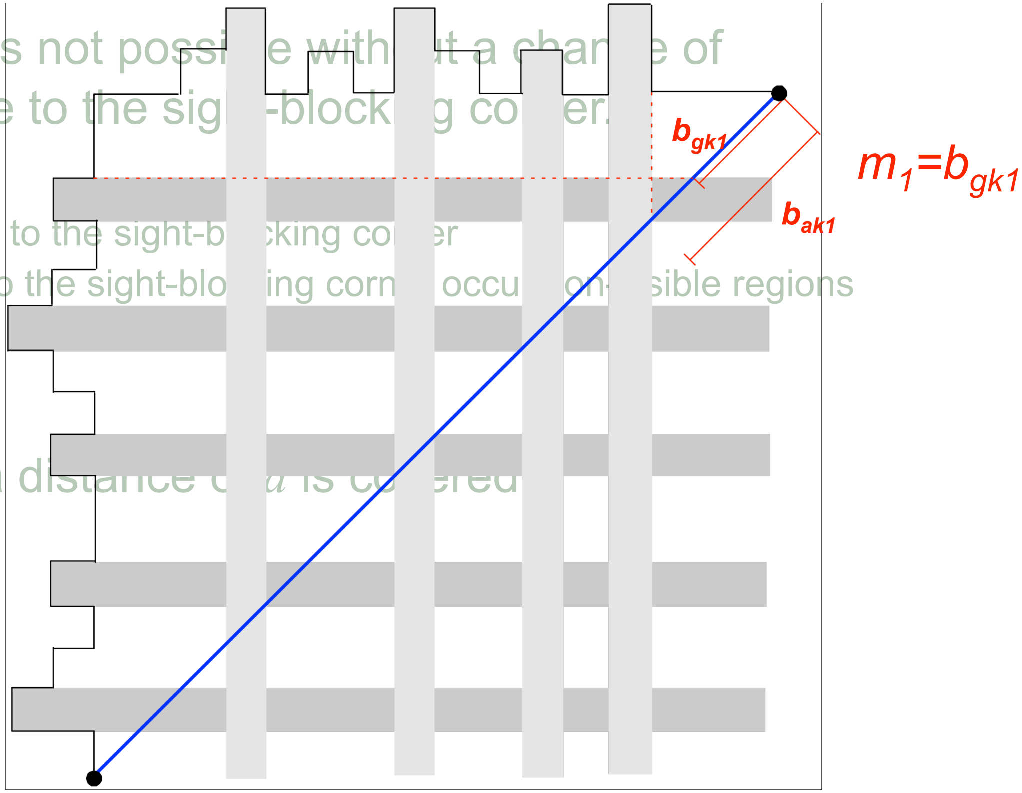
$a \leq 1$:

A. An axis-parallel move to E is possible without a turn
- e≥2a+1: interval case
- e<2a+1: extension case

B. An axis-parallel move to E is not possible without a change of direction: Let b_i be the distance to the sight-blocking corner.
- e ≥a+1: interval case
  - No non-visible region up to the sight-blocking corner
  - Along the boundary up to the sight-blocking corner occurs non-visible regions
- e<a+1: extension case

a > 1:

Similar; with scans every time a distance of a is covered



The strategy

$a \leq 1$:

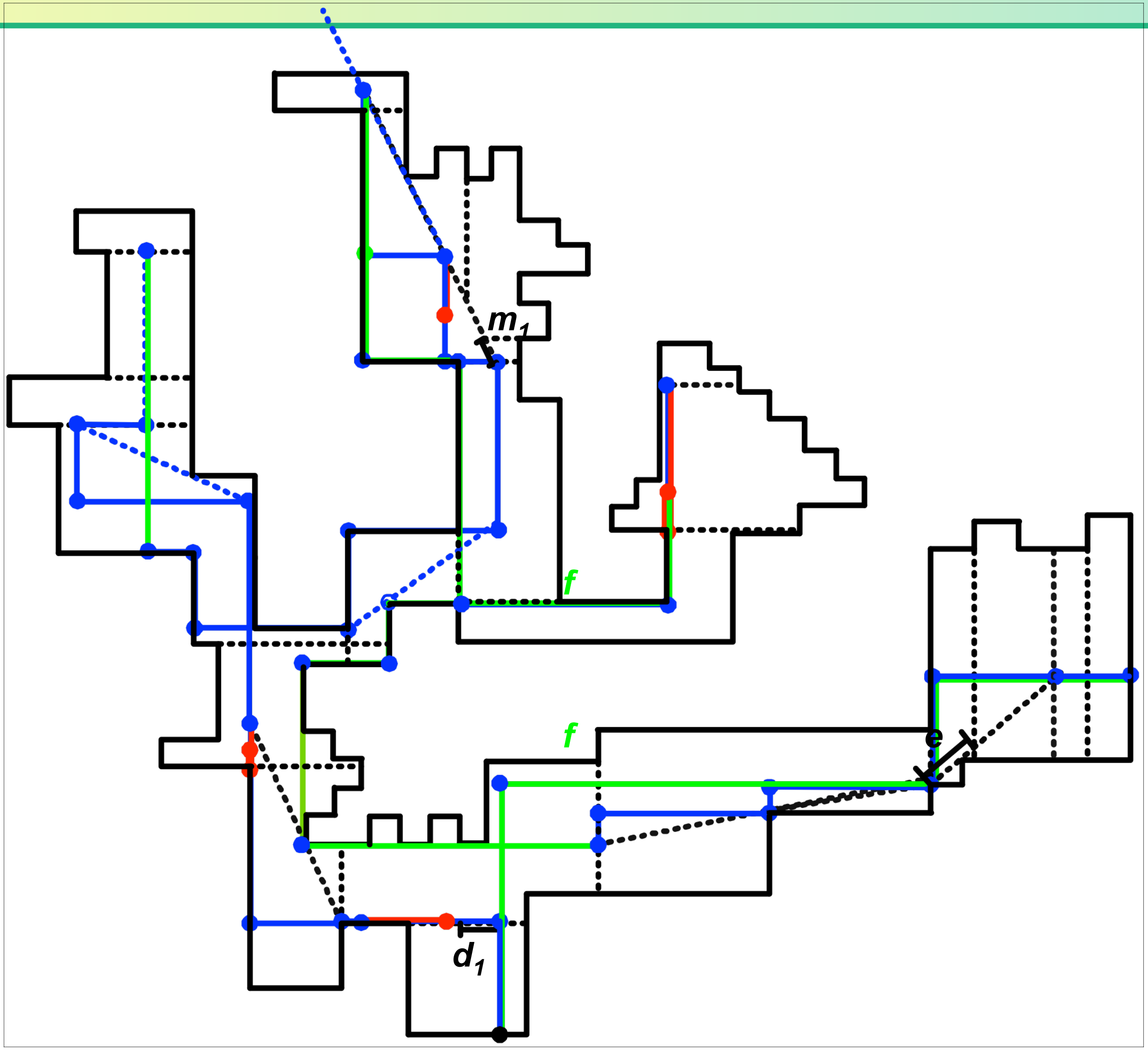
A. An axis-parallel move to E is possible without a turn
- e≥2a+1: interval case
- e<2a+1: extension case

B. An axis-parallel move to E is not possible without a change of direction: Let b_i be the distance to the sight-blocking corner.
- e ≥a+1: interval case
  - No non-visible region up to the sight-blocking corner
  - Along the boundary up to the sight-blocking corner occur non-visible regions
- e<a+1: extension case

$a > 1$:

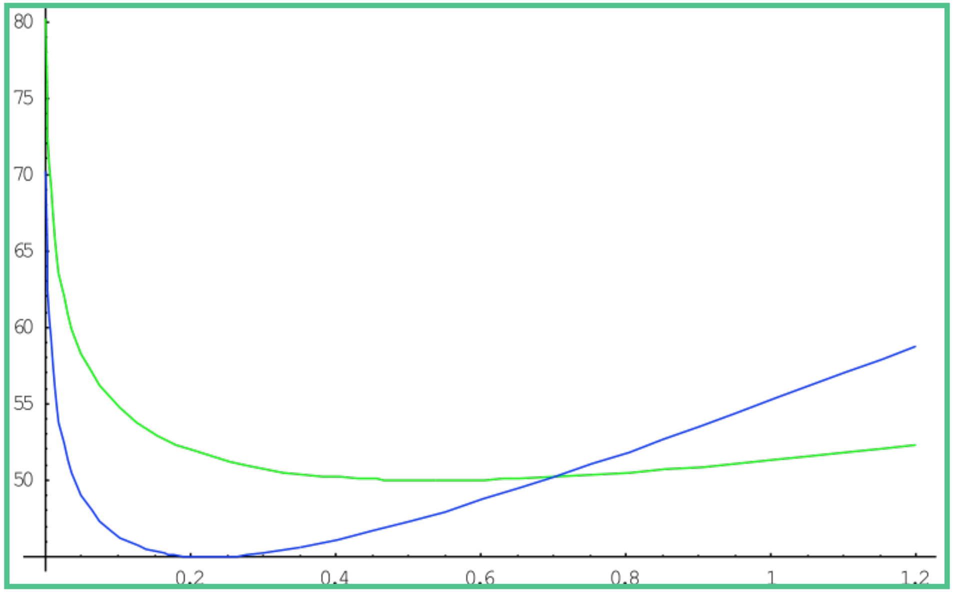
Similar; with scans every time a distance of a is covered.

An example



starting point

The competitive ratio of the strategy



a	upper bound for *c*
1	55.2294
0.8	51.8168
0.7	50.2083
0.5	50.0000
0.1	54.8000
0.01	67.0336
0.0001	93.4919
0.000001	120.0661

- If we assume $a = a_k$:

$$c \le \begin{cases} 8a + 34 + 4\dfrac{\ln\left(\dfrac{2a+3}{a}\right)}{\ln(2)}, & 0 \le a < 0.70043 \\ 20a + 24 + 4\dfrac{\ln\left(\dfrac{4a+3}{a}\right)}{\ln(2)}, & 0.70043 \le a \le 1 \end{cases}$$

Computational Geometry 43 (2010) 148–168

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

ELSEVIER

Polygon exploration with time-discrete vision

Sándor P. Fekete *, Christiane Schmidt [1]

Department of Computer Science, Technische Universität Braunschweig, D-38106 Braunschweig, Germany

A R T I C L E   I N F O

Article history:
Received 26 October 2008
Accepted 16 June 2009
Available online 21 June 2009

A B S T R A C T

With the advent of autonomous robots with two- and three-dimensional scanning capabilities, classical visibility-based exploration methods from computational geometry have gained in practical importance. However, real-life laser scanning of useful accuracy does not allow the robot to scan conti...

Part 1.3:
Searching with turn cost

Available online at www.sciencedirect.com

ScienceDirect

ELSEVIER

Theoretical Computer Science 361 (2006) 342–355

Theoretical Computer Science

www.elsevier.com/locate/tcs

Online searching with turn cost

Erik D. Demaine[a], Sándor P. Fekete[b,*], Shmuel Gal[c]

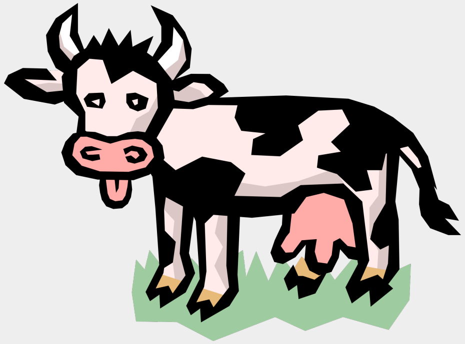
[a] *Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge MA, USA*

[b] *Department of Mathematical Optimization, Braunschweig University of Technology, Braunschweig, Germany*

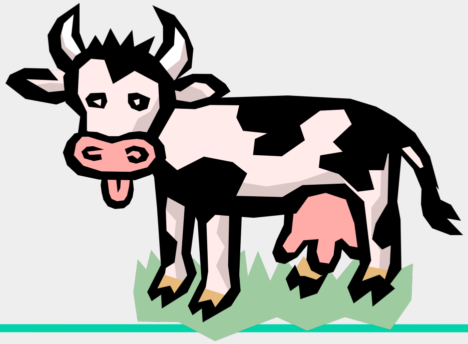
[c] *Department of Statistics, University of Haifa, Haifa, Israel*

Online Searching

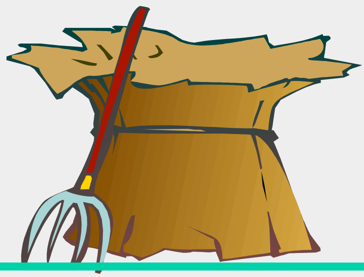
Online Searching

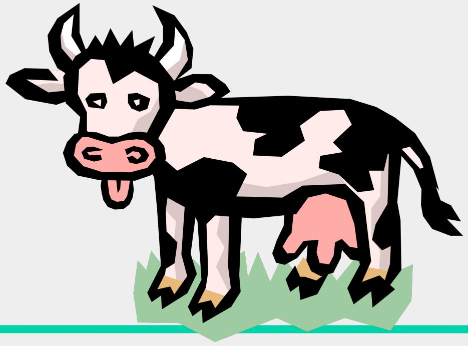


Online Searching

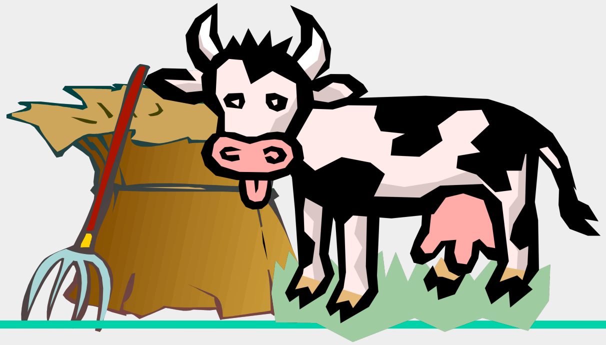


Online Searching

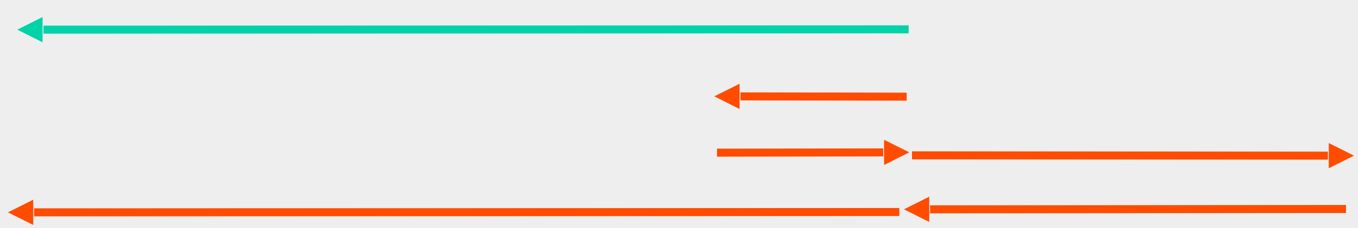




Online Searching



Online Searching



Linear Search

GIVEN: A starting position 0 on a line.

MISSION: Find an object at an unknown location.

UNKNOWN: (1) Direction of the object
(2) Distance OPT of the object

WANTED! A competitive strategy for the searcher that will guarantee that the object is found in time at most $c \cdot OPT$ for some constant "competitive" factor c.

Literature

BELLMAN 1963: Introduced the problem

BECK and NEWMAN 1970: Solved the problem

GAL 1974: Solved a generalization:

Search on m rays

Optimal competitive ratio: $1 + \frac{2m^m}{(m-1)^{m-1}}$

Optimal strategy: Geometric series with ratio $\left(\frac{m}{m-1}\right)$

Literature

KAO Also known as the cow-path problem

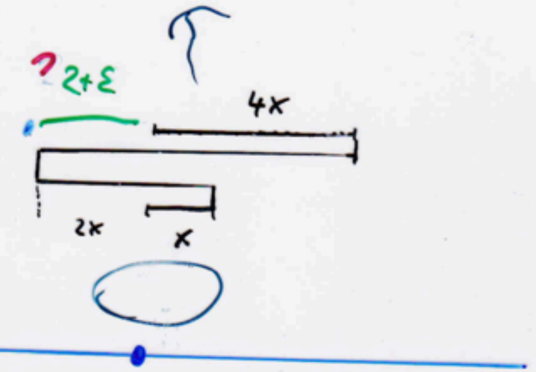
GAL 1980: Optimal trajectory to this type of problem is always a geometric series

BAEZA-YATES, CULBERSON, RAWLINS 1988: (and various others independently) Rediscovered problem and solution

Many variations and applications, in particular for geometric searching.

Doubling

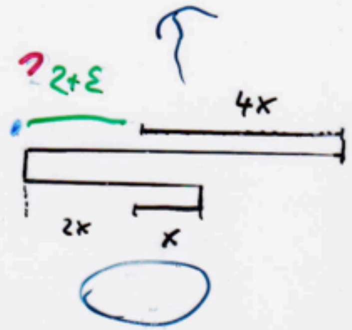
Keep doubling the search distance before returning:



KNOWN: This guarantees a competitive factor of 9, and this is best possible!

Doubling

Keep doubling the search distance before returning:



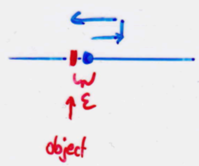
DISADVANTAGE: There is no real "start" of the trajectory - it's just a geometric series, and each previous step was half as long as the latest one!

Turn Cost

Immediate implications:

(1) There has to be a first move.

(2) A competitive factor is no longer possible:

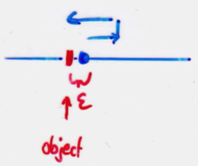


Turn Cost

Immediate implications:

(1) There has to be a first move.

(2) A competitive factor is no longer possible:



Searching in the wrong direction takes at least one turn, for a cost of d, compared to optimal ε

Fix: Consider $c \cdot OPT + f(d)$

- and possibly $c \cdot OPT + \lambda \cdot d$

An Open Problem

It is worth noting that the worst possible outcome of using the search strategy $s_{\bar{a}}$ ($\bar{a} \doteq 3.6$) is a loss of

$$1 + 2\sum_{j=-\infty}^{1} \bar{a}^j \doteq 10.9,$$

while the expected cost of the strategy s_2, which uses only minimax trajectories ($a = 2$), is $1 + 3/\ln 2 \doteq 5.3$. Thus, use of $s_{\bar{a}}$ yields (the minimal) expected cost of 4.6 but risks a maximal cost of 10.9, while use of s_2, which yields an expected cost of 5.3, minimizes the maximal cost (which in this case is equal to 9). The expected cost of any search strategy s_a with $2 < a < \bar{a}$ lies between 4.6 and 5.3, while the maximal cost lies between 9 and 10.9. All the strategies s_a with the parameter a lying outside the segment $[2, \bar{a}]$ are dominated by the family $\{s_a : 2 \le a \le \bar{a}\}$ with respect to the expected and the maximal cost.

8.4 Search with a Turning Cost

In this section we consider a more realistic version of the LSP, which has not been considered before in the literature. In this model the time spent in changing the direction of moving is not 0, as is usually assumed in the LSP, but a constant $d > 0$. Here, any search trajectory with a finite expected search time must have a first step because starting with an infinite number of oscillations takes infinite time. Therefore, assume for convenience that the search trajectory starts by going to $x_0 > 0$, then turning and going to $-x_1$, then turning and going to x_2, etc. (We can obviously assume that the searcher always goes with his maximal speed, 1, as is always the case with an immobile hider.) Thus

$$S = \{x_i\}_{i=0}^{\infty},$$

and denote

$$y_i = x_i + \frac{d}{2}, \quad i = 1, 2, \ldots.$$

In this case the normalized cost function (in the worst case) is not bounded near 0. Thus the reasonable cost function is the time to reach the target, $C(S, H)$, under the restriction $E|H| \le \lambda$. For convenience we assume $\lambda = 1$. Thus we are interested in

$$\hat{V} = \inf_S \sup_{h:E|H|\le 1} c(S, h).$$

We shall show that

$$9 + d \le \hat{V} \le 9 + 2d. \qquad (8.13)$$

The left inequality follows from equality (8.7), which implies that for any S and any δ, there always exist an x_i, as large as desired, with

$$\frac{2\sum_{j=0}^{i+1} y_j + x_i}{x_i} \sim \frac{2\sum_{j=0}^{i+1} y_j + y_i}{y_i} > 9 - \delta.$$

Thus, if the hider chooses h as

$$H = \begin{cases} -\varepsilon \text{ with probability } 1 - \frac{1}{x_i} & \text{and} \\ x_i + \varepsilon \text{ with probability } \frac{1}{x_i}, \end{cases}$$

then $E|H| \approx 1$ and, for a large enough x_i

$$c(S, h) \approx (2x_0 + d + \varepsilon)\left(1 - \frac{1}{x_i}\right) + \left(2\sum_{j=0}^{i+1} y_j + x_i\right)\frac{1}{x_i} \ge 9 - \delta + d$$

with $\delta > 0$ arbitrarily small.

In order to prove the right inequality of (8.13) we present a trajectory S that satisfies for all $x_I < |H| \le x_{I+2}$:

$$C(S, H) \le 9x_I + 2d \le 9|H| + 2d$$

so that for any h with $E|H| \le 1$

$$c(S, h) \le 9 + 2d.$$

We use the following approach. For any real γ, a sufficient condition for $v(S) \le 9+\gamma$ is the condition

$$\text{for all } |H| = x_i(+\varepsilon): \quad C(S, H) \le 9x_i + \gamma(+\varepsilon),$$

which will hold if the following conditions hold:

$$2\sum_{0}^{i+1} y_j = 8\left(y_i - \frac{d}{2}\right) + \gamma, \quad i = 0, 1, \ldots \qquad (8.14)$$

$$2y_0 = \gamma, \quad (\gamma > d/2)$$

$$y_i \ge d/2, \quad i = 0, 1, \ldots$$

Equality (8.14) is equivalent to (denoting $\frac{\gamma}{2} = b + 2d$)

$$y_{i+1} = 3y_i - \sum_{j=0}^{i-1} y_i + b, \quad i = 0, 1, \ldots \qquad (8.15)$$

$$y_0 = b + 2d\left(= \frac{\gamma}{2}\right)$$

$$y_i > d/2, \quad i = 0, 1 \ldots.$$

We now look for the minimal b which satisfies (8.15). It turns out that the general solution of (8.15) is

$$y_i = (y_0 + i\beta)2^i, \qquad (8.16)$$

An Open Problem

where $\beta \geq 0$ is a nonnegative parameter. (Because by (8.15) $y_{i+1} - y_i = 3y_i - 4y_{i-1}$, denoting $y_i = 2^i \alpha_i$ it easily follows that $\alpha_{i+1} - \alpha_i = \alpha_i - \alpha_{i-1}$, which leads to (8.16).)

Using (8.16) for $i = 0, 1$ in (8.15) it follows that $\beta = y_0 - d$. Since $\beta \geq 0$ and $\gamma = 2y_0$, it easily follows that $\gamma \geq 2d$. On the other hand, the value $9 + 2d$ can be achieved by the following trajectory

$$y_i = d2^i, \qquad x_i = d2^i - d/2, \quad i = 0, 1, \ldots$$

with the time to reach $x_i + \varepsilon$ being (neglecting $O(\varepsilon)$)

$$2\sum_{0}^{i+1} y_i + x_i = 2d(2^{i+2} - 1) + d2^i - d/2 = 9x_i + 2d.$$

Since $E|H| \leq 1$, the last equation guarantees expected time not exceeding $9 + 2d$.

Is $9 + 2d$ the best possible constant? This is still an open problem. (Note that (8.14) is a sufficient but not a necessary condition.)

Positions

The factor c can be at best 9!

(→ Consider d arbitrarily small compared to OPT.)

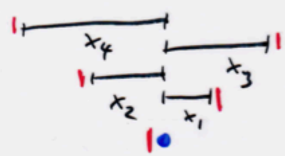
Suppose the searcher moves

x_1 to the right and returns,

x_2 to the left and returns,

x_3 to the right

(etc.)



Positions

The factor c can be at best 9!

(→ Consider d arbitrarily small compared to OPT.)

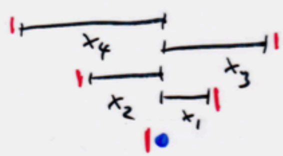
Suppose the searcher moves

x_1 to the right and returns,

x_2 to the left and returns,

x_3 to the right

(etc.)

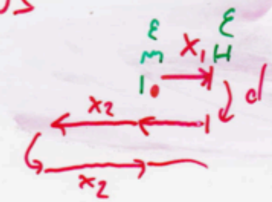


Critical positions for hiding:

$$y_0 = -\varepsilon$$
$$y_1 = x_1 + \varepsilon$$
$$y_2 = -x_2 - \varepsilon$$
$$y_3 = x_3 + \varepsilon$$

(etc.)

More Conditions



y_0 must be reached in time:

$$2x_1 + d + \varepsilon \le 9\varepsilon + \lambda d$$

y_1 must be reached in time:

$$2x_1 + 2x_2 + 2d + x_1 + \varepsilon \le 9(x_1+\varepsilon) + \lambda d$$

y_2 :

$$2x_1 + 2x_2 + 2x_3 + 3d + x_2 + \varepsilon \le 9(x_2+\varepsilon) + \lambda d$$

y_n :

$$2x_1 + \ldots + 2x_{n+1} + (n+1)d \le 8x_n + \lambda d$$

This must hold for all $\varepsilon > 0$, so we get

An Infinite LP

$$\min \lambda$$

$$\begin{aligned}
2x_1 &+ d \le \lambda d \\
2x_1 + 2x_2 &+ 2d \le 8x_1 + \lambda d \\
2x_1 + 2x_2 + 2x_3 &+ 3d \le 8x_2 + \lambda d \\
&\vdots \\
2x_1 + 2x_2 + 2x_3 + \dots + 2x_{n+1} &+ (n+1)d \le 8x_n + \lambda d \\
&\vdots \\
x_i &\ge 0
\end{aligned}$$

(1) Infinite primal optimal solution describes optimal strategy of searcher.

(2) Optimal λ is tight value of turn cost penalty.

(3) Infinite dual optimal solution gives explicit proof of tightness.

Solving the Infinite LP

Solving Subsystems

Only using the first n constraints yields a relaxation, with solutions $x_i^{(n)}$ and λ_n.
Each λ_n is a lower bound for λ.

Approach:

(1) Run CPLEX on subsystems

(2) Consider convergence of solutions

(3) Construct infinite solution

(4) Verify solution

Solutions

Solutions

n	λ_n	$x_1^{(n)}$	$x_2^{(n)}$	$x_3^{(n)}$	$x_4^{(n)}$	$c_1^{(n)}$	$c_2^{(n)}$	$c_3^{(n)}$	$c_4^{(n)}$
1	1.0000	0.0000				1.0000			
2	1.2500	0.1250	0.0000			0.7500	0.2500		
3	1.4166	0.2083	0.3333	0.0000		0.6666	0.2500	0.0833	
4	1.5313	0.2656	0.5625	0.6875	0.000	0.6250	0.2500	0.0937	0.0312
5	1.6125	0.3062	0.7250	1.1750	1.3000	0.6000	0.2500	0.1000	0.0375
10	1.8001	0.4000	1.1003	2.3001	4.3031	0.5500	0.2500	0.1125	0.0500
20	1.9000	0.4500	1.3000	2.9000	5.900	0.5250	0.2500	0.1187	0.0562
40	1.9500	0.4750	1.4000	3.2000	6.4333	0.5125	0.2500	0.1218	0.0593
50	1.9600	0.4800	1.4200	3.2600	6.8600	0.5100	0.2500	0.1225	0.0600
100	1.9800	0.4900	1.4600	3.3800	7.1600	0.5050	0.2500	0.1237	0.0612
200	1.9900	0.4950	1.4800	3.4400	7.3400	0.5025	0.2500	0.1243	0.0618
400	1.9950	0.4975	1.4900	3.4700	7.4200	0.5012	0.2500	0.1245	0.0621

Solutions

n	λ_n	$x_1^{(n)}$	$x_2^{(n)}$	$x_3^{(n)}$	$x_4^{(n)}$	$c_1^{(n)}$	$c_2^{(n)}$	$c_3^{(n)}$	$c_4^{(n)}$
1	1.0000	0.0000				1.0000			

Table 1
Solutions for a number of linear subsystems

n	λ_n	$x_1^{(n)}$	$x_2^{(n)}$	$x_3^{(n)}$	$x_4^{(n)}$	$x_5^{(n)}$
1	1.0000	0.0000				
2	1.2500	0.1250	0.0000			
3	1.4166	0.2083	0.3333	0.0000		
4	1.5312	0.2656	0.5625	0.6875	0.0000	
5	1.6125	0.3062	0.7250	1.1750	1.3000	0.0000
6	1.6718	0.3359	0.8437	1.5312	2.2500	2.3750
7	1.7165	0.3582	0.9330	1.7991	2.9642	4.1607
8	1.7509	0.3754	1.0019	2.0058	3.5156	5.5930
9	1.7782	0.3891	1.0563	2.1692	3.9130	6.6284
10	1.8001	0.4000	1.1003	2.3011	4.3031	7.5078
20	1.9000	0.4500	1.3000	2.9000	5.9000	11.5000
30	1.9333	0.4666	1.3666	3.1000	6.4333	12.8333
40	1.9500	0.4750	1.4000	3.2000	6.7000	13.5000
50	1.9600	0.4800	1.4200	3.2600	6.8600	13.9000
100	1.9800	0.4900	1.4600	3.3800	7.1800	14.7000
200	1.9900	0.4950	1.4800	3.4400	7.3400	15.1000
400	1.9950	0.4975	1.4900	3.4700	7.4200	15.3000

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
200	1.9900	0.4950	1.4800	3.4400	7.3400	0.5025	0.2500	0.1243	0.0618
400	1.9950	0.4975	1.4900	3.4700	7.4200	0.5012	0.2500	0.1245	0.0621

Solutions

n	λ_n	$x_1^{(n)}$	$x_2^{(n)}$	$x_3^{(n)}$	$x_4^{(n)}$	$c_1^{(n)}$	$c_2^{(n)}$	$c_3^{(n)}$	$c_4^{(n)}$
1	1.0000	0.0000				1.0000			

Table 1
Solutions for a number of linear subsystems

n	λ_n	$x_1^{(n)}$	$x_2^{(n)}$	$x_3^{(n)}$	$x_4^{(n)}$	$x_5^{(n)}$
1	1.0000	0.0000				
2	1.2500	0.1250	0.0000			
3	1.4166	0.2083	0.3333	0.0000		
4	1.5312	0.2656	0.5625	0.6875	0.0000	
5	1.6125	0.3062	0.7250	1.1750	1.3000	0.0000
6	1.6718	0.3359	0.8437	1.5312	2.2500	2.3750
7	1.7165	0.3582	0.9330	1.7991	2.9642	4.1607
8	1.7509	0.3754	1.0019	2.0058	3.5156	5.5930
9	1.7782	0.3891	1.0563	2.1692	3.9130	6.6284
10	1.8001	0.4000	1.1003	2.3011	4.3031	7.5078
20	1.9000	0.4500	1.3000	2.9000	5.9000	11.5000
30	1.9333	0.4666	1.3666	3.1000	6.4333	12.8333
40	1.9500	0.4750	1.4000	3.2000	6.7000	13.5000
50	1.9600	0.4800	1.4200	3.2600	6.8600	13.9000
100	1.9800	0.4900	1.4600	3.3800	7.1800	14.7000
200	1.9900	0.4950	1.4800	3.4400	7.3400	15.1000
400	1.9950	0.4975	1.4900	3.4700	7.4200	15.3000

[illegible]00	1.[illegible]800	0.4[illegible]00	1.4600	[illegible]	[illegible]	0.5050	0.2500	0.125[illegible]	0.061[illegible]
200	1.9900	0.4950	1.4800	3.4400	7.3400	0.5025	0.2500	0.1243	0.0618
400	1.9950	0.4975	1.4900	3.4700	7.4200	0.5012	0.2500	0.1245	0.0621
∞	2.0000	0.5000	1.5000	3.5000	7.5000	0.5000	0.2500	0.1250	0.0625

Table 1
Solutions for a number of linear subsystems

n	λ_n	$x_1^{(n)}$	$x_2^{(n)}$	$x_3^{(n)}$	$x_4^{(n)}$	$x_5^{(n)}$
1	1.0000	0.0000				
2	1.2500	0.1250	0.0000			
3	1.4166	0.2083	0.3333	0.0000		
4	1.5312	0.2656	0.5625	0.6875	0.0000	
5	1.6125	0.3062	0.7250	1.1750	1.3000	0.0000
6	1.6718	0.3359	0.8437	1.5312	2.2500	2.3750
7	1.7165	0.3582	0.9330	1.7991	2.9642	4.1607
8	1.7509	0.3754	1.0019	2.0058	3.5156	5.5930
9	1.7782	0.3891	1.0563	2.1692	3.9130	6.6284
10	1.8001	0.4000	1.1003	2.3011	4.3031	7.5078
20	1.9000	0.4500	1.3000	2.9000	5.9000	11.5000
30	1.9333	0.4666	1.3666	3.1000	6.4333	12.8333
40	1.9500	0.4750	1.4000	3.2000	6.7000	13.5000
50	1.9600	0.4800	1.4200	3.2600	6.8600	13.9000
100	1.9800	0.4900	1.4600	3.3800	7.1800	14.7000
200	1.9900	0.4950	1.4800	3.4400	7.3400	15.1000
400	1.9950	0.4975	1.4900	3.4700	7.4200	15.3000

10	1.8001	0.4000	1.1003	2.3001	4.3031	0.5500	0.2500	0.1125	0.0500
20	1.9000	0.4500	1.3000	2.9000	5.900	0.5250	0.2500	0.1187	0.0562
40	1.9500	0.4750	1.4000	3.2000	6.4333	0.5125	0.2500	0.1218	0.0593
50	1.9600	0.4800	1.4200	3.2600	6.8600	0.5100	0.2500	0.1225	0.0600
100	1.9800	0.4900	1.4600	3.3800	7.1800	0.5050	0.2500	0.1237	0.0612
200	1.9900	0.4950	1.4800	3.4400	7.3400	0.5025	0.2500	0.1243	0.0618
400	1.9950	0.4975	1.4900	3.4700	7.4200	0.5012	0.2500	0.1245	0.0621
∞	2.0000	0.5000	1.5000	3.5000	7.5000	0.5000	0.2500	0.1250	0.0625

Table 1
Solutions for a number of linear subsystems

n	λ_n	$x_1^{(n)}$	$x_2^{(n)}$	$x_3^{(n)}$	$x_4^{(n)}$	$x_5^{(n)}$
1	1.0000	0.0000				
2	1.2500	0.1250	0.0000			
3	1.4166	0.2083	0.3333	0.0000		
4	1.5312	0.2656	0.5625	0.6875	0.0000	
5	1.6125	0.3062	0.7250	1.1750	1.3000	0.0000
6	1.6718	0.3359	0.8437	1.5312	2.2500	2.3750
7	1.7165	0.3582	0.9330	1.7991	2.9642	4.1607
8	1.7509	0.3754	1.0019	2.0058	3.5156	5.5930
9	1.7782	0.3891	1.0563	2.1692	3.9130	6.6284
10	1.8001	0.4000	1.1003	2.3011	4.3031	7.5078
20	1.9000	0.4500	1.3000	2.9000	5.9000	11.5000
30	1.9333	0.4666	1.3666	3.1000	6.4333	12.8333
40	1.9500	0.4750	1.4000	3.2000	6.7000	13.5000
50	1.9600	0.4800	1.4200	3.2600	6.8600	13.9000
100	1.9800	0.4900	1.4600	3.3800	7.1800	14.7000
200	1.9900	0.4950	1.4800	3.4400	7.3400	15.1000
400	1.9950	0.4975	1.4900	3.4700	7.4200	15.3000

10 1.8001 0.4000 1.1003 2.3001 4.3031 0.5500 0.2500 0.1125 0.0500

n	λ_n	$y_1^{(n)}$	$y_2^{(n)}$	$y_3^{(n)}$	$y_4^{(n)}$	$y_5^{(n)}$
1	1.0000					
2	1.2500	0.7500	0.2500			
3	1.4166	0.6666	0.2500	0.0833		
4	1.5312	0.0625	0.2500	0.0937	0.0312	
5	1.6125	0.6000	0.2500	0.1000	0.0375	0.0125
6	1.6718	0.5833	0.2500	0.1041	0.0416	0.0156
7	1.7165	0.5714	0.2500	0.1071	0.0446	0.0178
8	1.7509	0.5625	0.2500	0.1093	0.0468	0.0195
9	1.7782	0.5555	0.2500	0.1111	0.0486	0.0208
10	1.8001	0.5500	0.2500	0.1125	0.0500	0.0218
20	1.9000	0.5250	0.2500	0.1187	0.0562	0.0265
30	1.9333	0.5166	0.2500	0.1208	0.0583	0.0281
40	1.9500	0.5125	0.2500	0.1218	0.0593	0.0289
50	1.9600	0.5100	0.2500	0.1225	0.0600	0.0293
100	1.9800	0.5050	0.2500	0.1237	0.0612	0.0303
200	1.9900	0.5025	0.2500	0.1243	0.0618	0.0307
400	1.9950	0.5012	0.2500	0.1245	0.0621	0.0310

Verifying the Solution

Choose: $x_i = \left(2^i - \frac{1}{2}\right) d$

$c_j = \frac{1}{2^j}$

Check primal solution, i.e. search strategy:

Verifying the Solution

Choose:

$$x_i = \left(2^i - \tfrac{1}{2}\right) d$$

$$c_j = \frac{1}{2^j}$$

Check primal solution, i.e. search strategy:

Inequality n yields

$$\sum_{i=1}^{n+1} 2(x_i) - 8x_n + (n+1)d \leq \lambda d$$

or

$$\sum_{i=1}^{n+1} 2\left(2^i - \tfrac{1}{2}\right)d - 8\left(2^{n-1} - \tfrac{1}{2}\right)d + (n+1)d \leq \lambda d$$

or

$$2^{n+2} - 2 \quad - 2^{n+2} + 4 \leq \lambda$$

or

$$2 \leq \lambda$$

So we have a feasible solution with $\lambda = 2$.

Verifying the Dual

$$\min \lambda$$

$$
\begin{array}{llllll}
2x_1 & & & & +d & \leq \lambda d \\
2x_1 & +2x_2 & & & +2d & \leq 8x_1 + \lambda d \\
2x_1 & +2x_2 & +2x_3 & & +3d & \leq 8x_2 + \lambda d \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
2x_1 & +2x_2 & +2x_3 & \;\; +2x_{n+1} & +(n+1)d & \leq 8x_n + \lambda d \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 & & & & x_i & \geq 0
\end{array}
$$

Verifying the Dual

$$\min \lambda$$

$$\begin{array}{lll}
2x_1 & +d & \leq \lambda d \\
2x_1 + 2x_2 & +2d & \leq 8x_1 + \lambda d \\
2x_1 + 2x_2 + 2x_3 & +3d & \leq 8x_2 + \lambda d \\
\vdots & \vdots & \vdots \\
2x_1 + 2x_2 + 2x_3 \; \cdots \; +2x_{n+1} & +(n+1)d & \leq 8x_n + \lambda d \\
\vdots & \vdots & \vdots \\
 & x_i & \geq 0
\end{array}$$

Consider infinite linear combination of with the dual multipliers:

The resulting coefficient of x_n is

$$\sum_{i=n}^{\infty} \frac{2}{2^i} - \frac{8}{2^{n+1}} = 0$$

The resulting coefficient of λd is

$$\sum_{i=1}^{\infty} \frac{1}{2^i} = 1$$

Verifying the Dual

$$\min \lambda$$

$$\begin{array}{llllll}
2x_1 & & & & +d & \leq \lambda d \\
2x_1 & +2x_2 & & & +2d & \leq 8x_1 + \lambda d \\
2x_1 & +2x_2 & +2x_3 & & +3d & \leq 8x_2 + \lambda d \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
2x_1 & +2x_2 & +2x_3 & \;\; +2x_{n+1} & +(n+1)d & \leq 8x_n + \lambda d \\
\vdots & \vdots & \vdots & \vdots \;\; \ddots & \vdots & \vdots \\
 & & & & x_i & \geq 0
\end{array}$$

Consider infinite linear combination of with the dual multipliers:

The resulting coefficient of x_n is

$$\sum_{i=n}^{\infty} \frac{2}{2^i} - \frac{8}{2^{n+1}} = 0$$

The resulting coefficient of λd is

$$\sum_{i=1}^{\infty} \frac{1}{2^i} = 1$$

This leaves the inequality

$$\sum_{i=1}^{\infty} i \left(\tfrac{1}{2}\right)^i d \leq \lambda d$$

Using $\sum_{i=1}^{\infty} i x^i = \frac{x}{(1-x)^2}$, this implies

$$2 \leq \lambda,$$

so we have an explicit lower bound.

More General Problem

Cow-Path Problem with Turn Cost

Scenario: m rays from the origin.

Turn cost on a ray: d_1

Turn cost at the origin: d_2

Total turn cost for changing from one ray to another: $d = d_1 + d_2$

Known: Asymptotic competitive ratio for $d = 0$ is

$$1 + \frac{2m^m}{(m-1)^{m-1}} =: 1 + M$$

Constraints

REWRITE CONSTRAINTS:

$$2 \sum_{j=1}^{n+m-1} x_i + (n+m-1)\, d \leq M x_n + \lambda d$$

AGAIN:
- Infinite LP for determining λ
- Run experiments for fixed m

Solving the Problem

Solution of the Problem

Here described: $m = 3$

$$\lambda_{1000} = 3.743996$$

$$x_1^{(1000)} = 0.2492495$$

$$x_2^{(1000)} = 0.6227485$$

$$x_3^{(1000)} = 1.182434$$

$$x_4^{(1000)} = 2.021118$$

$$x_5^{(1000)} = 3.277878$$

Solving the Problem

Solution of the Problem

Here described: $m = 3$

$$\lambda_{1000} = 3.743996$$
$$x_1^{(1000)} = 0.2492495$$
$$x_2^{(1000)} = 0.6227485$$
$$x_3^{(1000)} = 1.182434$$
$$x_4^{(1000)} = 2.021118$$
$$x_5^{(1000)} = 3.277878$$

After adjusting for logarithmic convergence:

$$\lambda = 3.75 = \frac{15}{4}$$
$$x_1 = 0.25 = \frac{1}{4}$$
$$x_2 = 0.625 = \frac{5}{8}$$

} educated guesses

Assuming all constraints are tight, we get a recursion for x_n, yielding:

$$x_3 = \frac{19}{16} = 1.1875 \quad ✓$$
$$x_4 = \frac{65}{32} = 2.03125 \quad ✓$$
$$x_5 = \frac{211}{64} = 3.296875 \quad ✓$$

Solution II

SOLUTION FOR $m=3$ (Cont.)

Using the structure of the recursion, we conclude

$$x_n = \frac{d}{2}\left(\left(\frac{3}{2}\right)^n - 1\right)$$

Not hard to check:

Together with $\lambda = \frac{15}{4}$, this satisfies all constraints with equality.

Dual Variables

$C_2^{(1000)} = 0.445339$

$C_3^{(1000)} = 0.1481481$

$C_4^{(1000)} = 0.1481481$

$C_5^{(1000)} = 0.08217275$

$C_6^{(1000)} = 0.06022488$

$C_7^{(1000)} = 0.038277$

$C_8^{(1000)} = 0.02610326$

Using (*), we get the recursive condition

$$C_n = \frac{27}{4}\left(C_{n+2} - C_{n+3}\right)$$

or

$$C_{n+3} = \frac{27}{4} C_{n+2} - C_n .$$

Some values:

$C_5 = \frac{60}{3^6} = 0.0823045$ ✓

$C_6 = \frac{132}{3^7} = 0.0603566$ ✓

$C_7 = \frac{252}{3^8} = 0.0384087$ ✓

$C_8 = \frac{516}{3^9} = 0.0262155$ ✓

Dual Routing

Explicit formula after solving recursion:

$$c_j = \frac{2^{j+1} + (-1)^j 4}{3^{j+1}}$$

Dual Routing

Dual Routing

VERIFYING THE DUAL

Consider the infinite linear combination of all constraints, using the computed c_j.

- By assumption, we have

$$\sum_{i=2}^{\infty} c_i = 1$$

so the coefficient of $2d$ is 1.

- By recursion, all coefficients of x_n cancel.

Dual Routing

– This leaves

$$\sum_{i=2}^{\infty} i c_i \leq \lambda$$

Using the explicit values of c_i and $\sum_{i=1}^{\infty} i x^i = \frac{x}{(1-x)^2}$,

we get

$$\lambda \geq \sum_{i=2}^{\infty} i c_i = \frac{2}{3} \sum_{i=1}^{\infty} i \left(\frac{2}{3}\right)^i + \frac{4}{3} \sum_{i=1}^{\infty} i \left(-\frac{1}{3}\right)^i$$

$$= \frac{2}{3} \frac{\frac{2}{3}}{\left(1-\frac{2}{3}\right)^2} + \frac{4}{3} \frac{-\frac{1}{3}}{\left(1+\frac{1}{3}\right)^2}$$

$$= 4 - \frac{1}{4} = \frac{15}{4} = 3.75$$

Dual Routing

$$\sum_{j=m-1}^{\infty} jy_j = \sum_{j=m-1}^{2m-2} jy_j + \sum_{j=2m-1}^{\infty} jy_j$$

$$= \sum_{j=m-1}^{2m-2} jy_j + \sum_{j=m-1}^{\infty} (j+m)y_{j+m}$$

$$= \sum_{j=m-1}^{2m-2} jy_j + \sum_{j=m-1}^{\infty} (j+m)\left(y_{j+m-1} - \frac{1}{M}y_j\right)$$

$$= (2m-2)y_{2m-2} + \sum_{j=m-1}^{2m-3} jy_j + \sum_{j=m-1}^{\infty} (j+m-1)y_{j+m-1}$$

$$+ \sum_{j=m-1}^{\infty} y_{j+m-1} - \sum_{j=m-1}^{\infty} \frac{1}{M}jy_j - \sum_{j=m-1}^{\infty} \frac{m}{M}y_j$$

$$= \frac{2m-2}{M} + \sum_{j=m-1}^{\infty} jy_j + \left(1 - \sum_{j=m-1}^{2m-3} y_j\right) - \sum_{j=m-1}^{\infty} \frac{1}{M}jy_j - \frac{m}{M},$$

hence

$$\sum_{j=m-1}^{\infty} jy_j = 2m - 2 + (M - m - (m-2)) - m = M - m,$$

as claimed. □

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 361 (2006) 342–355

Theoretical Computer Science

www.elsevier.com/locate/tcs

Online searching with turn cost

Erik D. Demaine[a], Sándor P. Fekete[b,*], Shmuel Gal[c]

[a] Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge MA, USA
[b] Department of Mathematical Optimization, Braunschweig University of Technology, Braunschweig, Germany
[c] Department of Statistics, University of Haifa, Haifa, Israel

Part 2: Several Robots

Asymptotics

Asymptotics

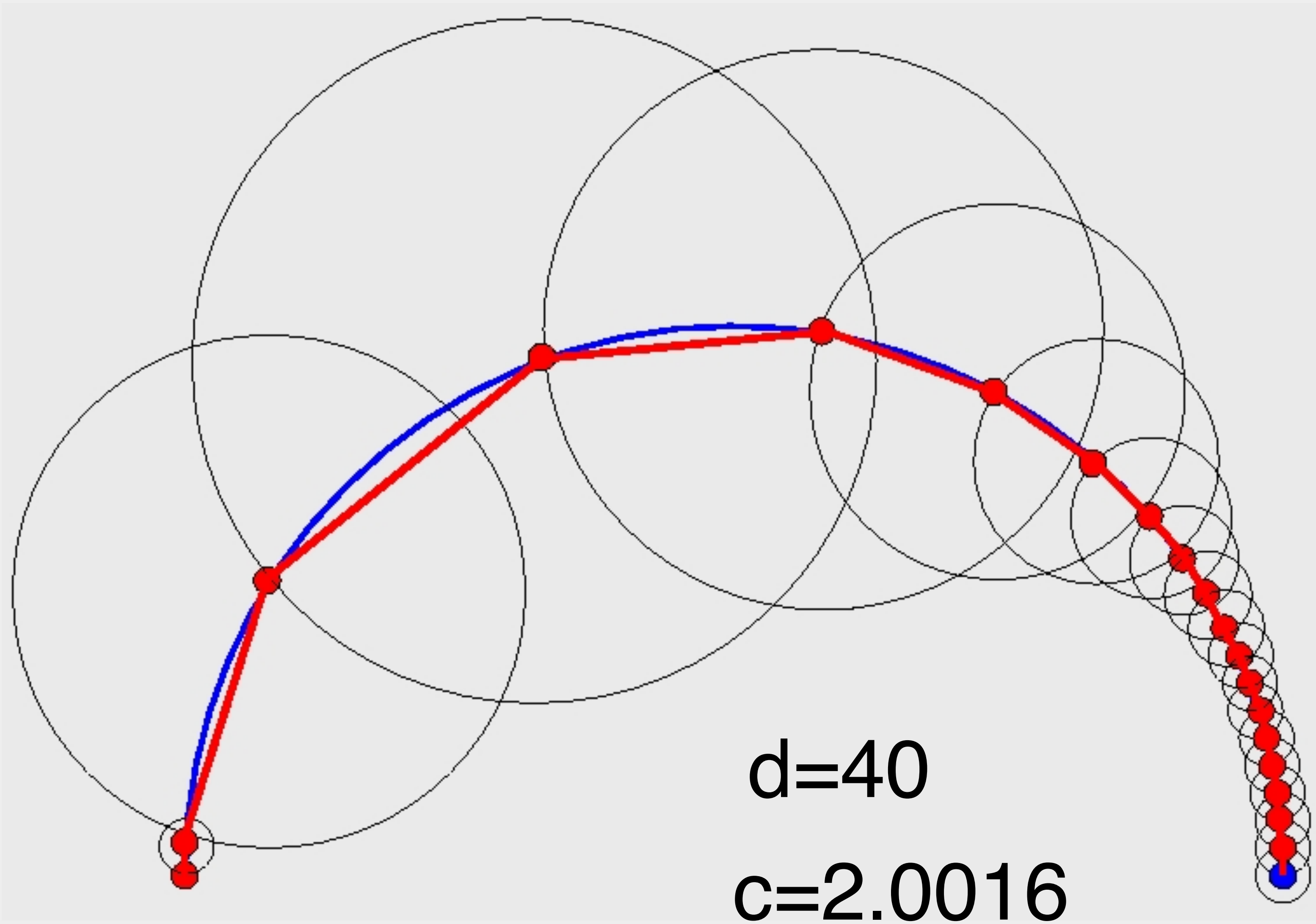
d=40

Asymptotics

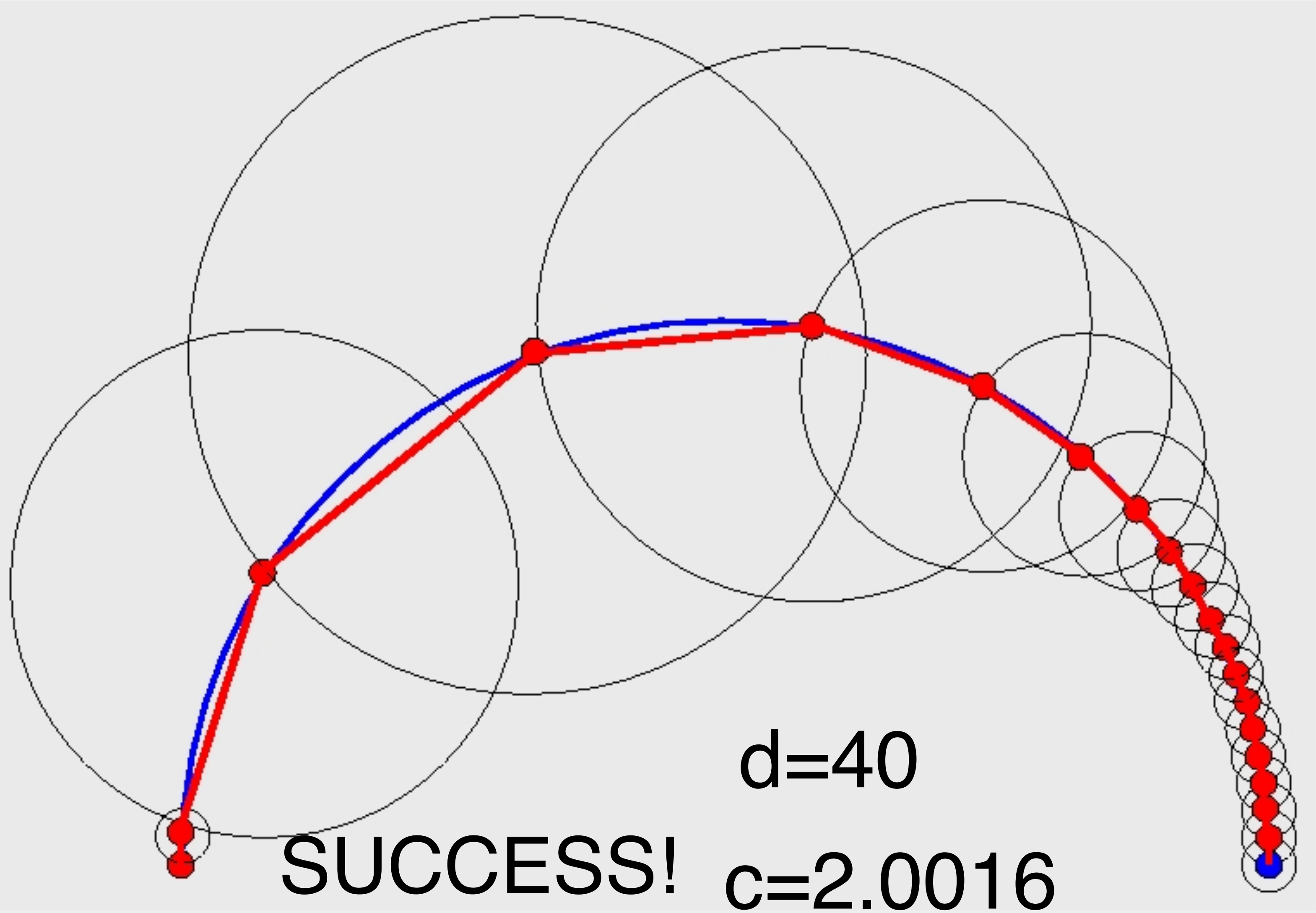
d=40

c=2.0016

Asymptotics



Asymptotics



Asymptotics

d=40

c=2.0016

Asymptotics

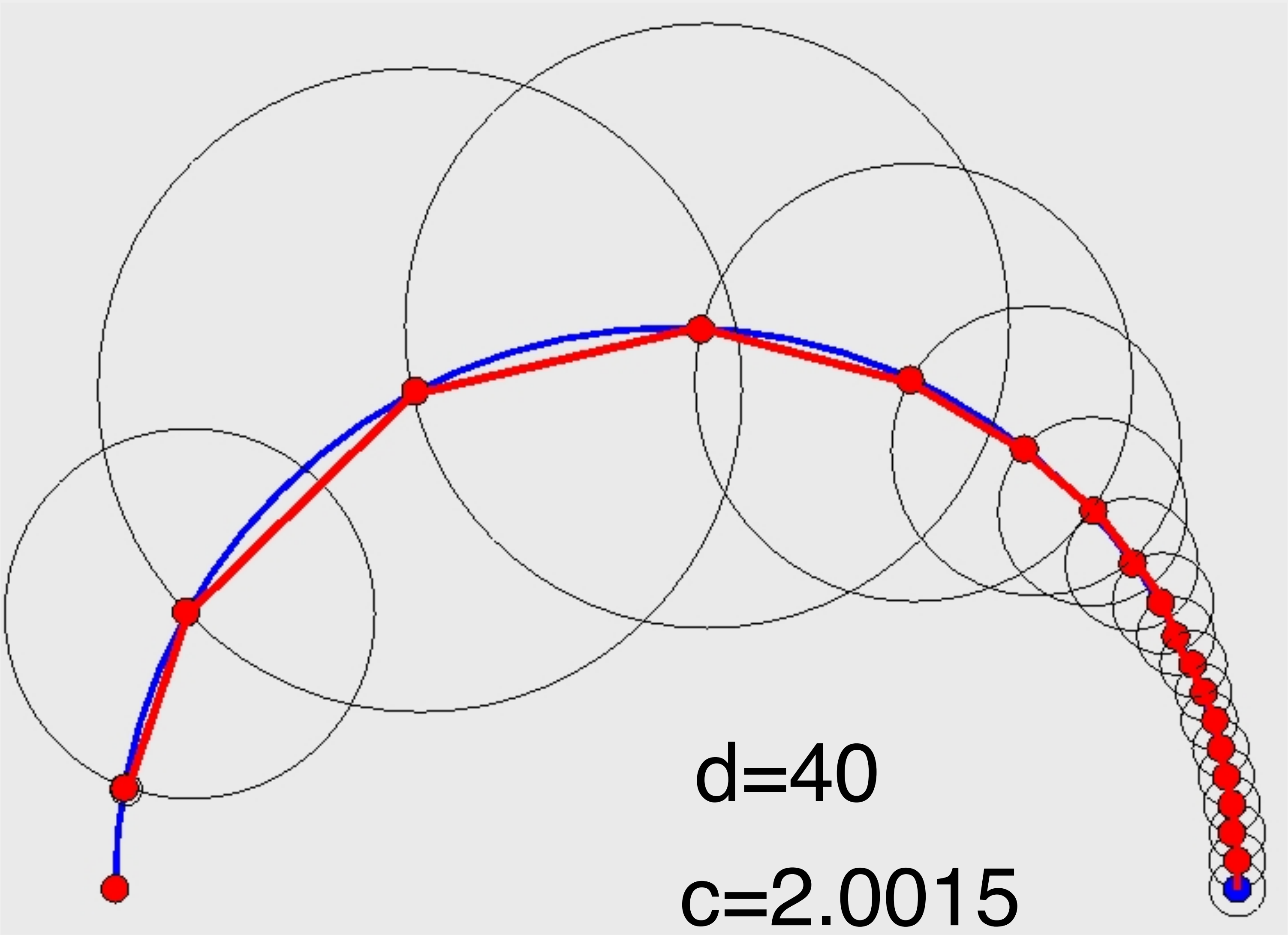
d=40

Asymptotics

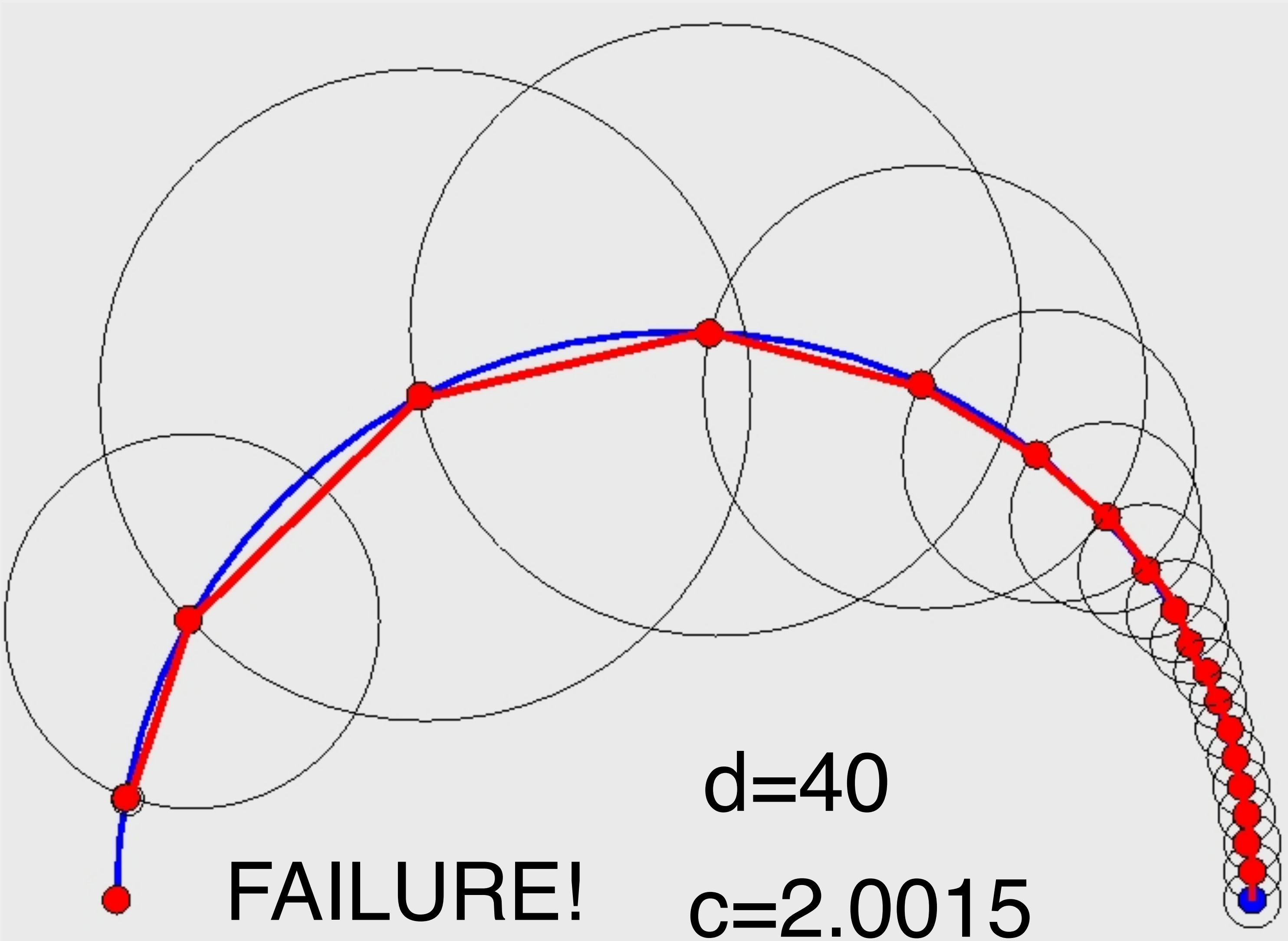
d=40

c=2.0015

Asymptotics

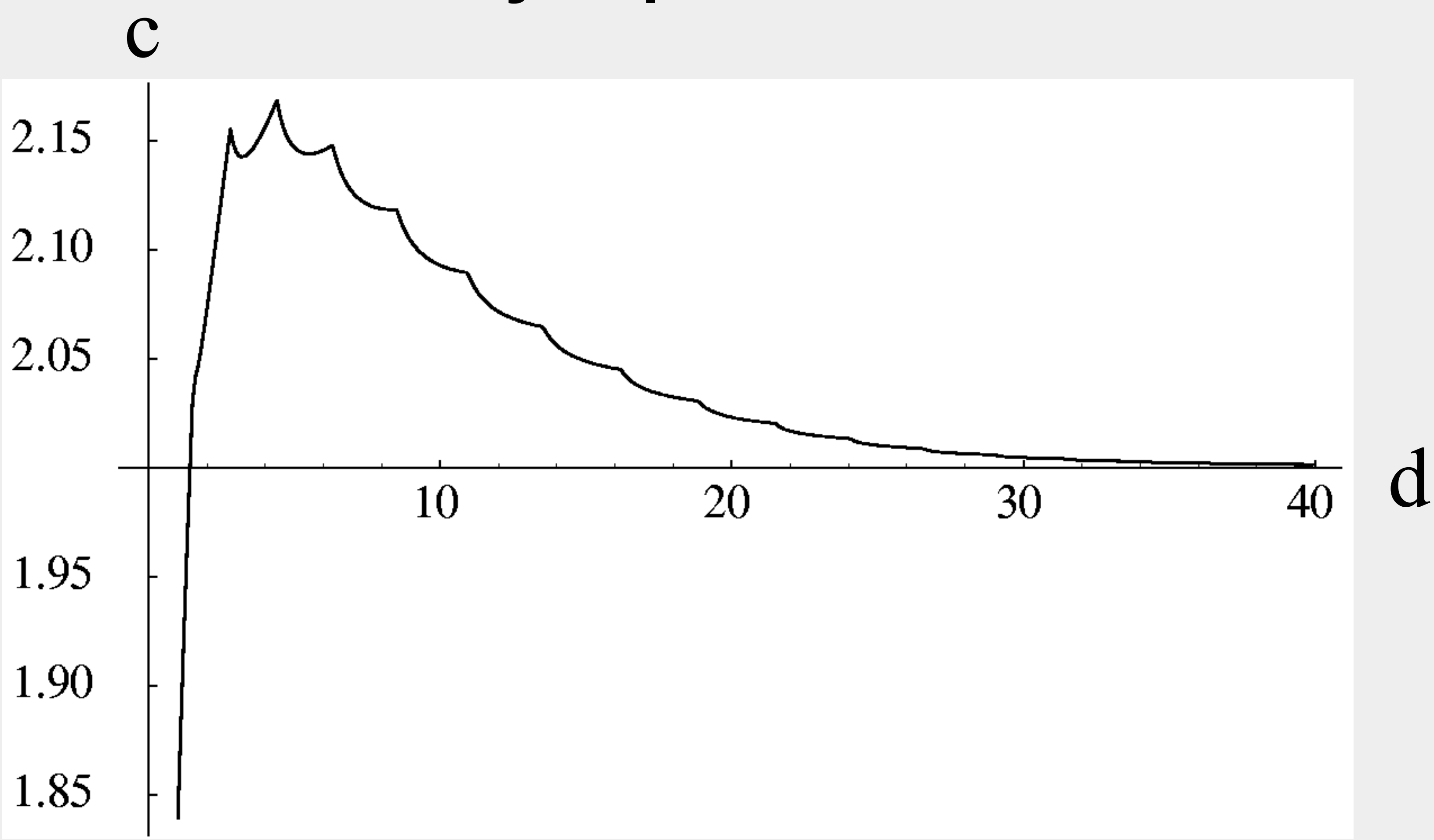


Asymptotics

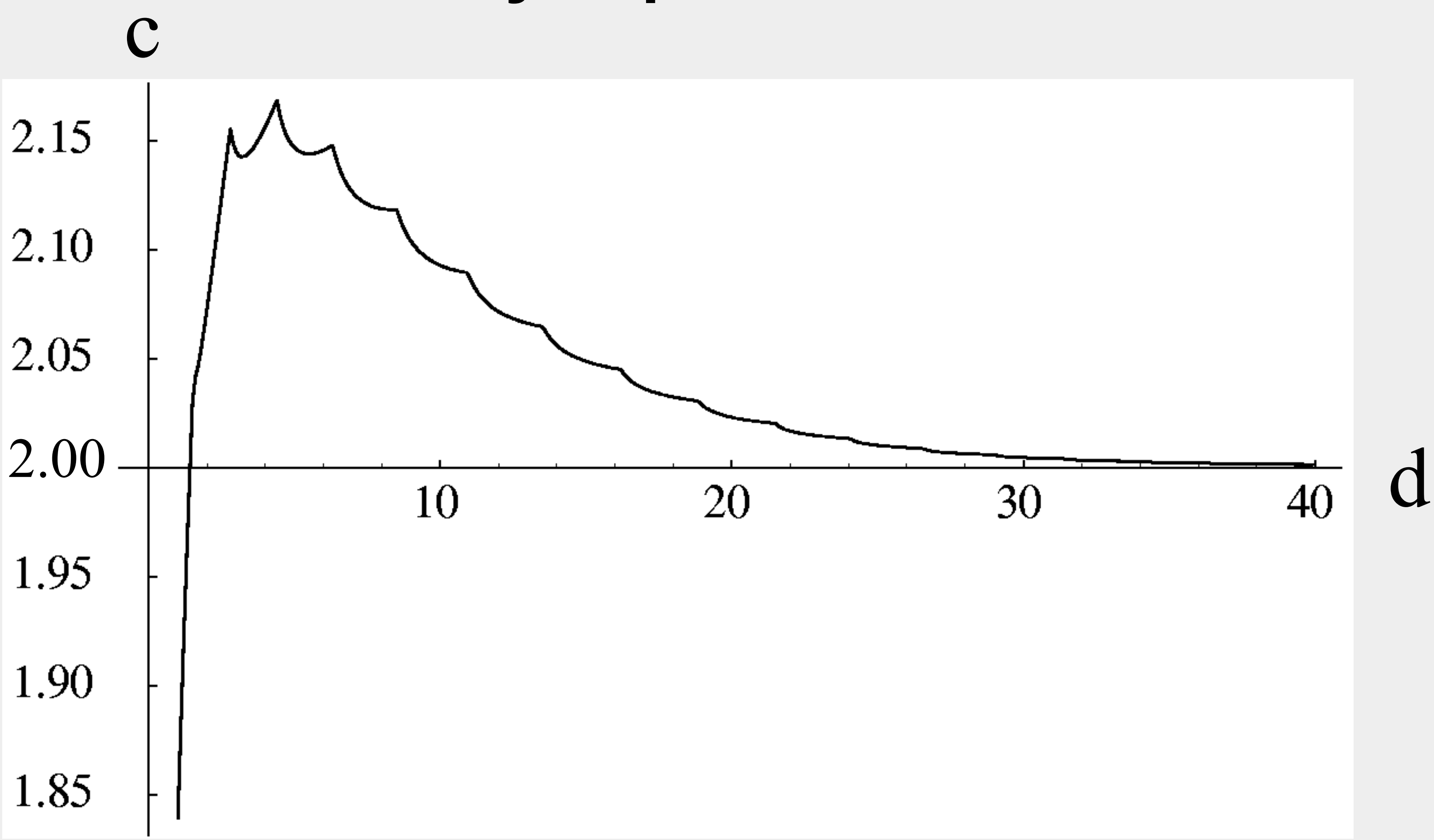


Asymptotics

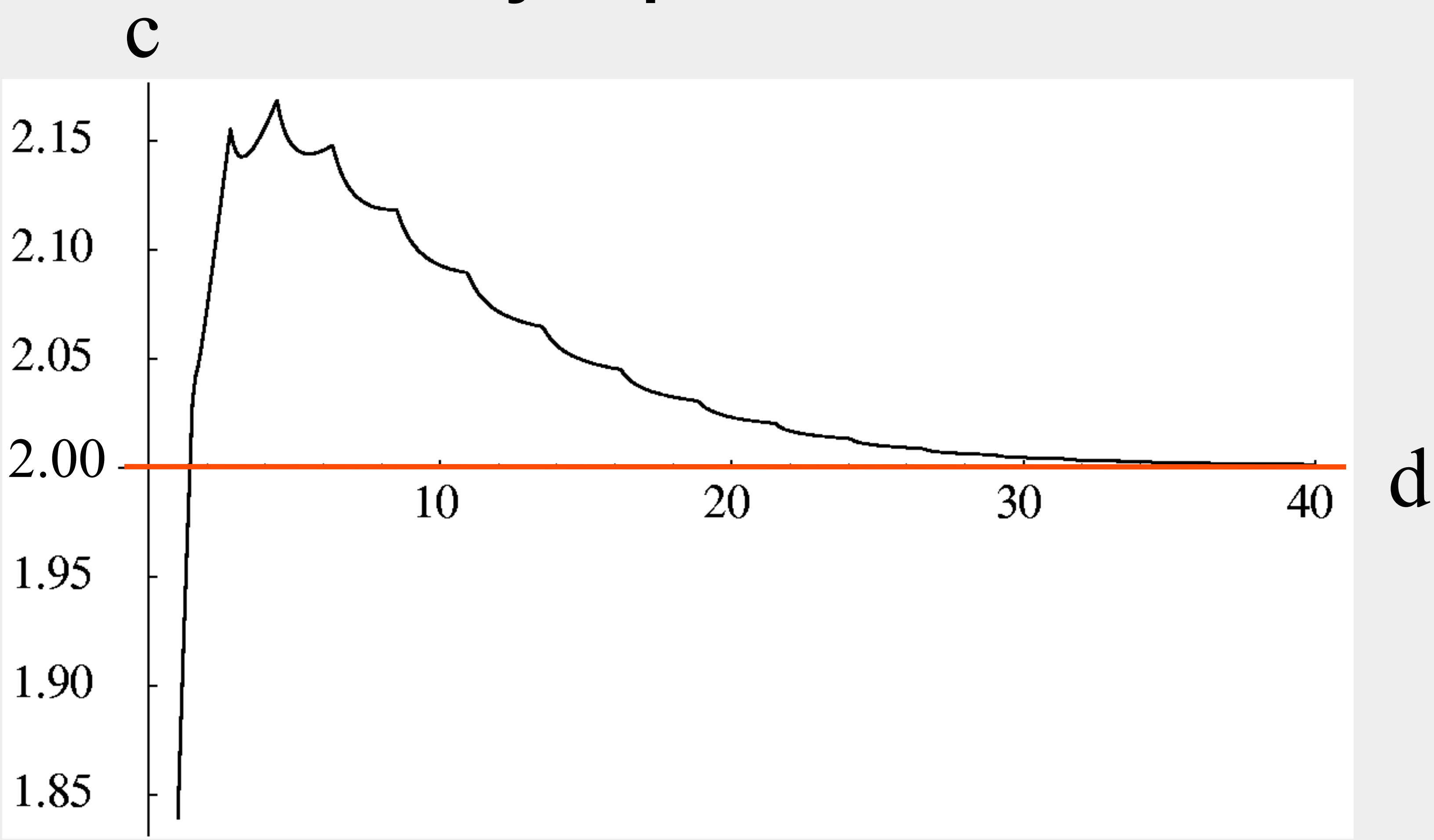
Asymptotics



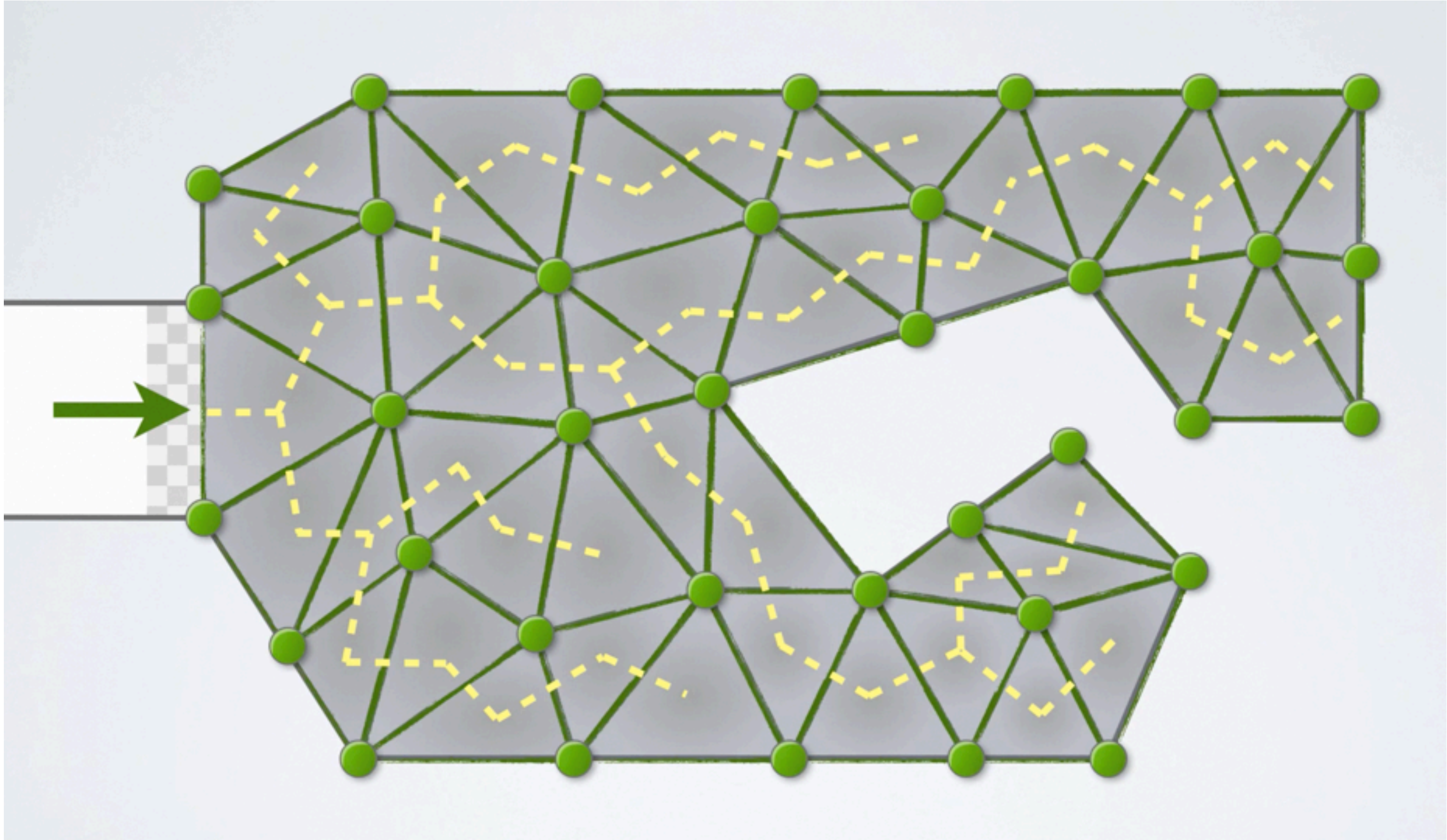
Asymptotics



Asymptotics



Collective Tree Exploration

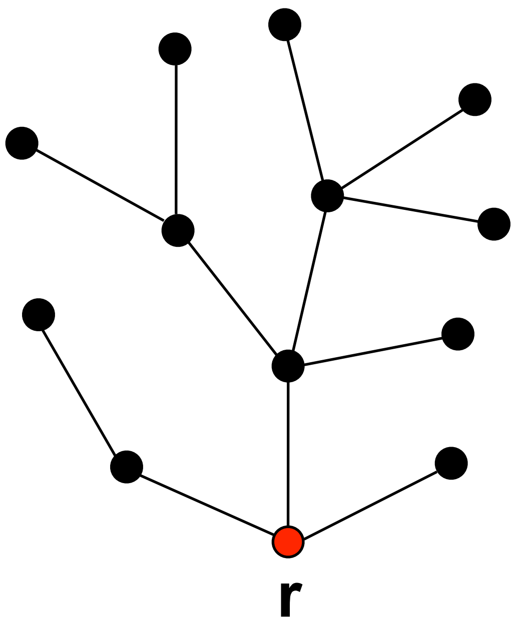


Tree Exploration

Tree Exploration

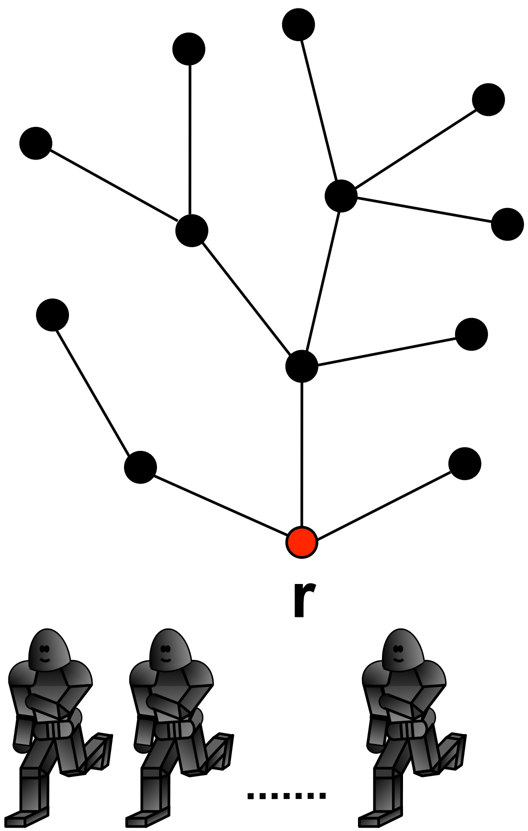
Given:
Unknown tree T, root r

Tree Exploration



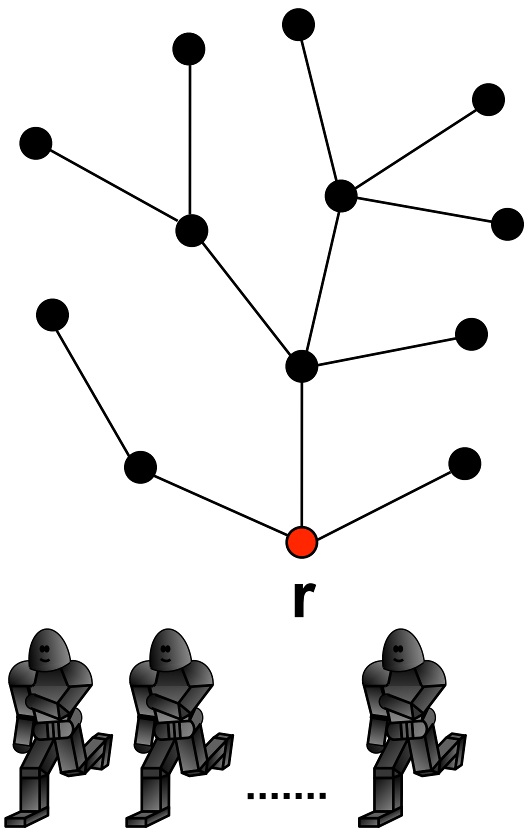
Given:
Unknown tree T, root r
k robots, initially located at r

Tree Exploration



Given:
Unknown tree T, root r
k robots, initially located at r

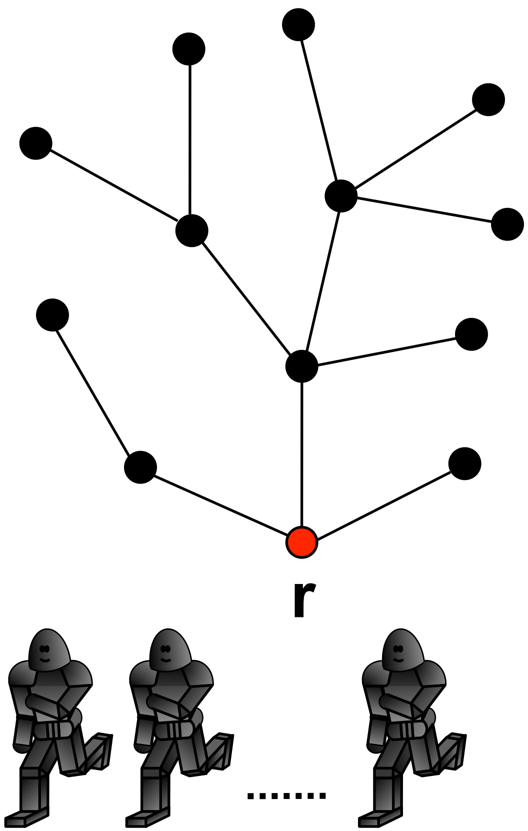
Tree Exploration



Given:
Unknown tree T, root r
k robots, initially located at r

Task:

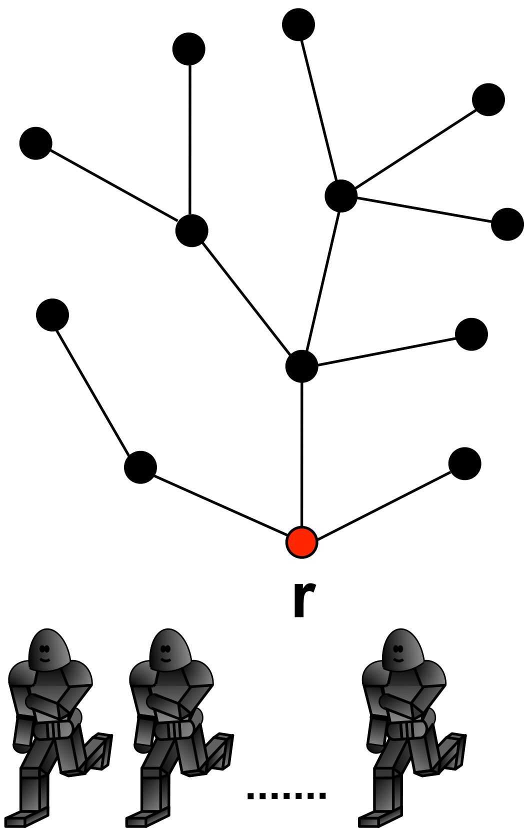
Tree Exploration



Given:
Unknown tree T, root r
k robots, initially located at r

Task:
Explore T and return to origin

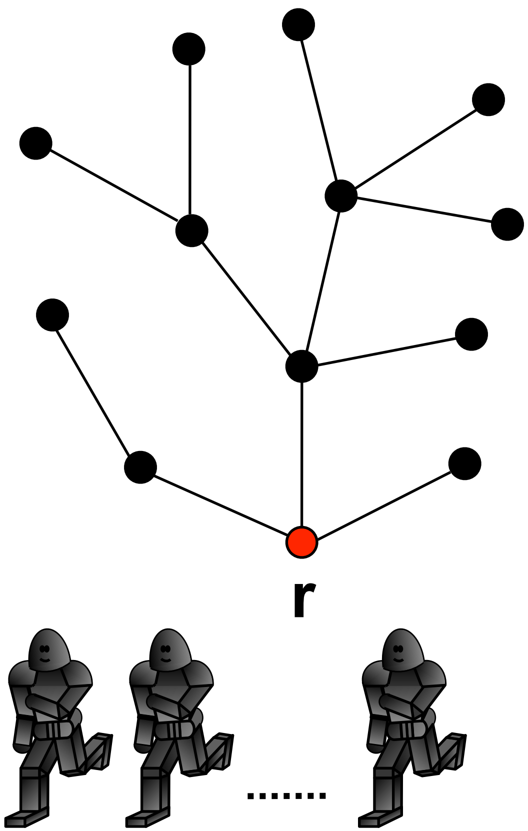
Tree Exploration



Given:
Unknown tree T, root r
k robots, initially located at r

Task:
Explore T and return to origin

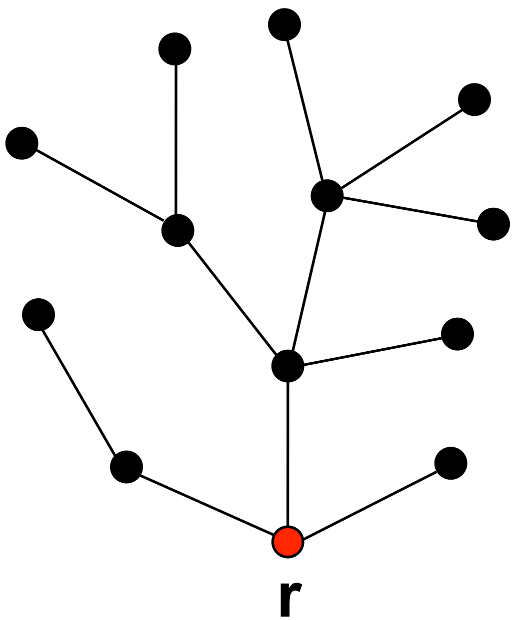
Tree Exploration

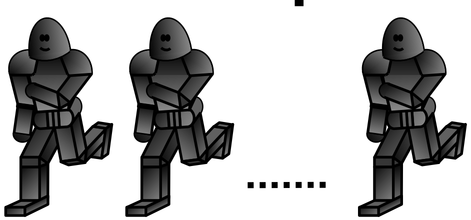


Given:
Unknown tree T, root r
k robots, initially located at r

Task:
Explore T and return to origin

Tree Exploration



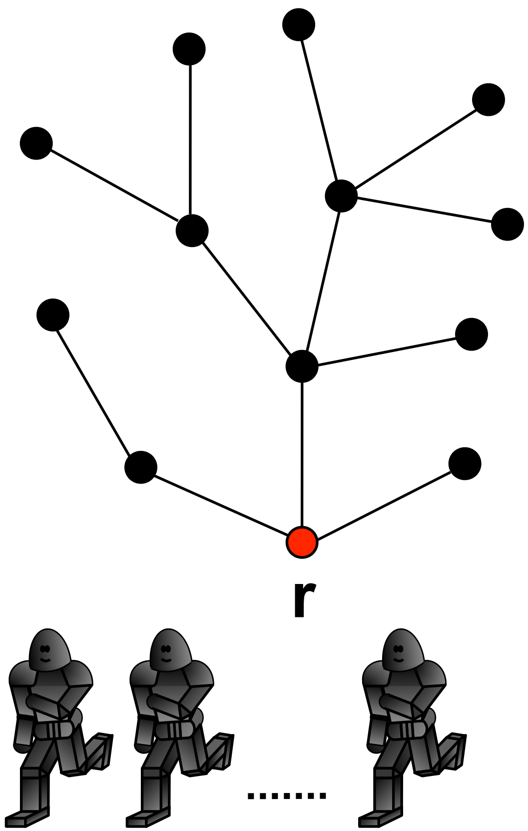


Given:
Unknown tree T, root r
k robots, initially located at r

Task:
Explore T and return to origin

Objective:

Tree Exploration



Given:
Unknown tree T, root r
k robots, initially located at r

Task:
Explore T and return to origin

Objective:
Minimize maximum workload

Previous Work

Previous Work

Dynia et al. (2006):

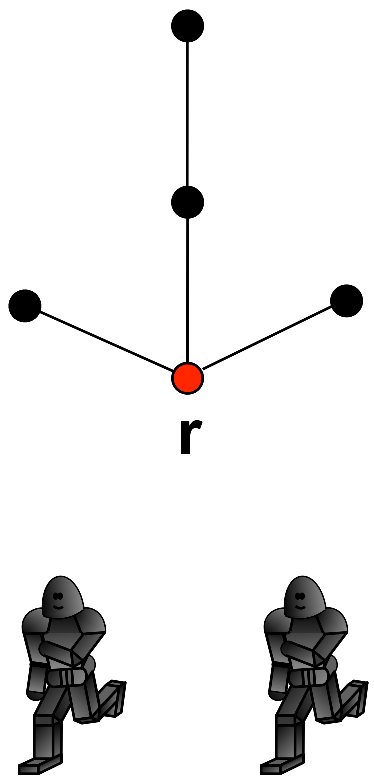
- Lower bound of 3/2 on

Previous Work

Dynia et al. (2006):

- Lower bound of 3/2 on competitive factor
- An appropriate greedy algorithm achieves competitive factor of 8

Previous Work

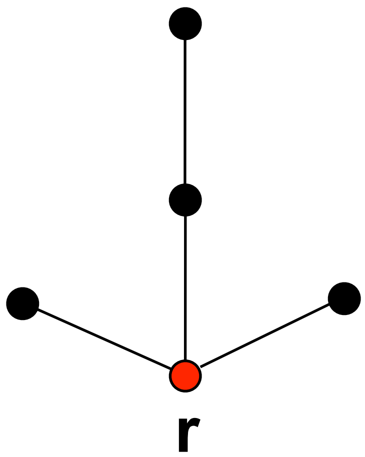


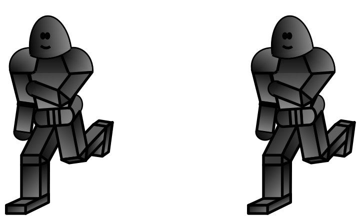
Dynia et al. (2006):

- Lower bound of 3/2 on competitive factor
- An appropriate greedy algorithm achieves competitive factor of 8

Previous Work

$$\frac{ALG}{OPT} = \frac{6}{4} = \frac{3}{2}$$



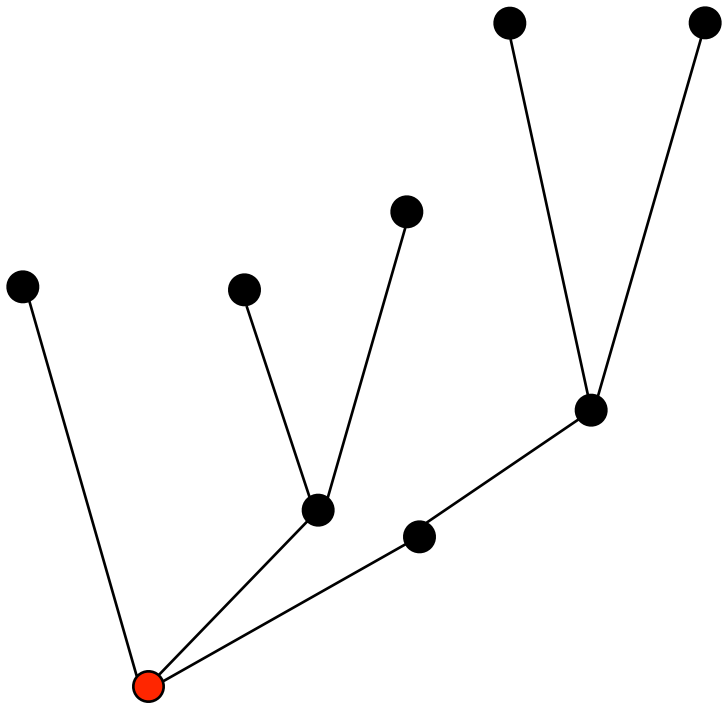


Dynia et al. (2006):

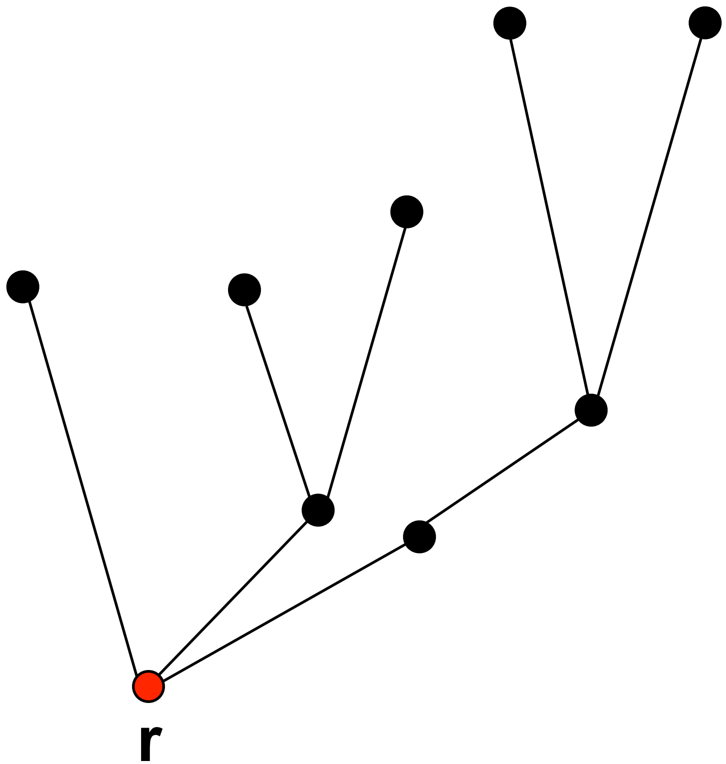
- Lower bound of 3/2 on competitive factor
- An appropriate greedy algorithm achieves competitive factor of 8

A New Strategy for General Trees

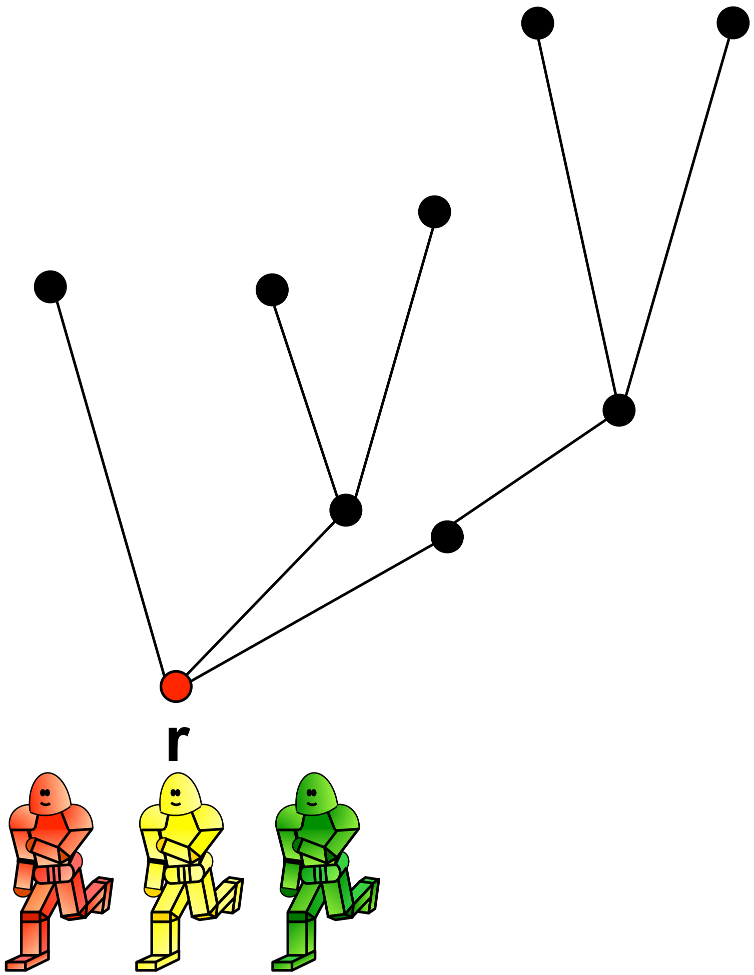
A New Strategy for General Trees



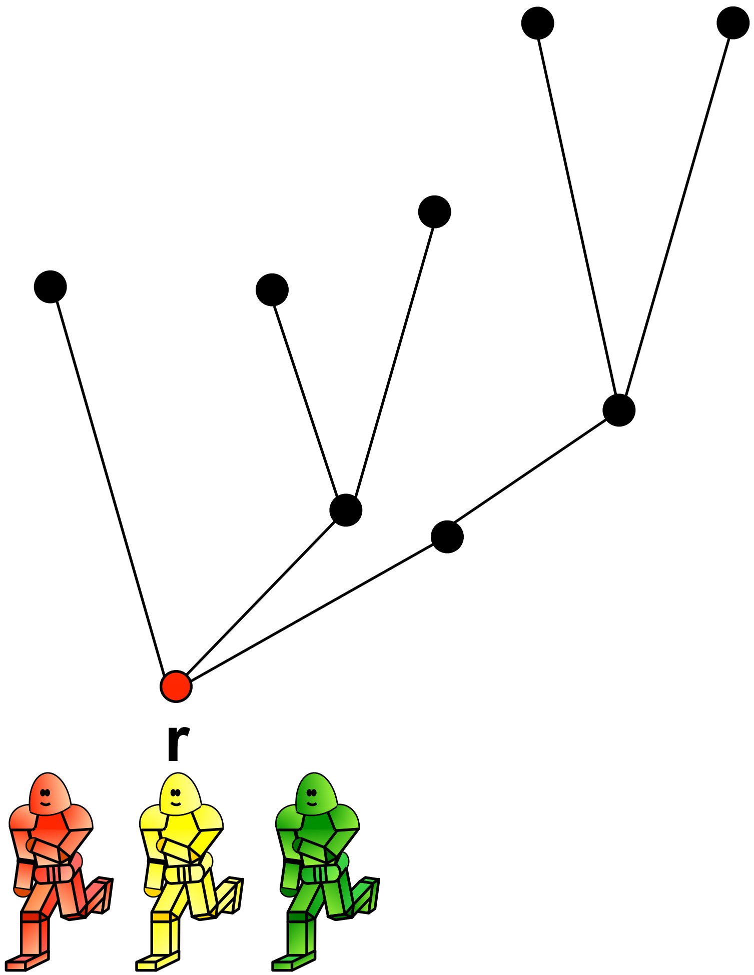
A New Strategy for General Trees



A New Strategy for General Trees

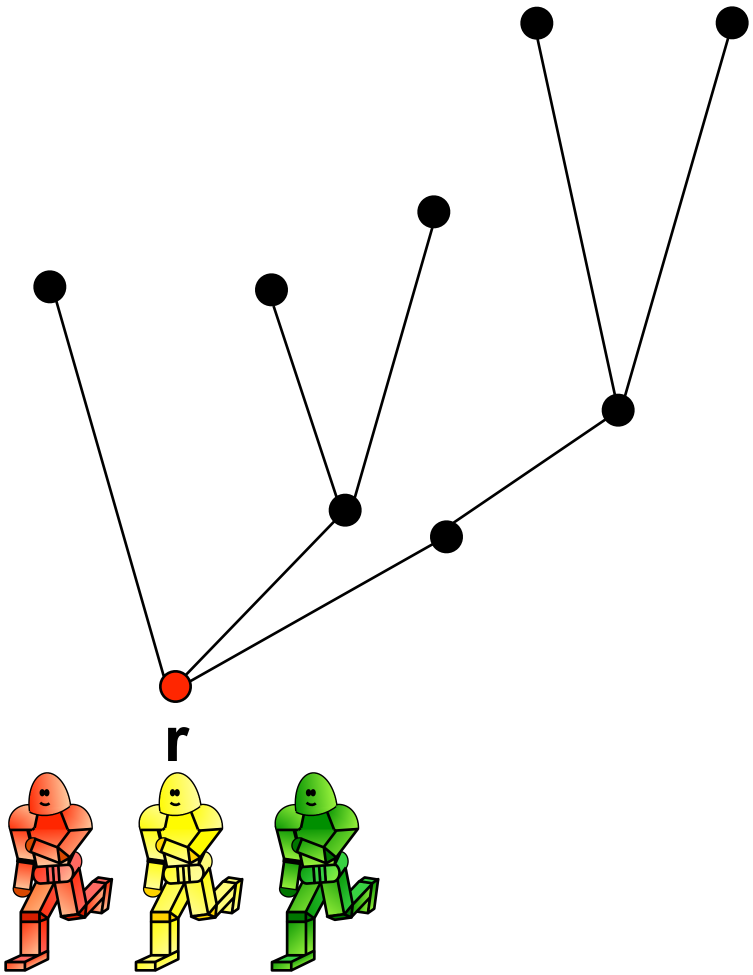


A New Strategy for General Trees



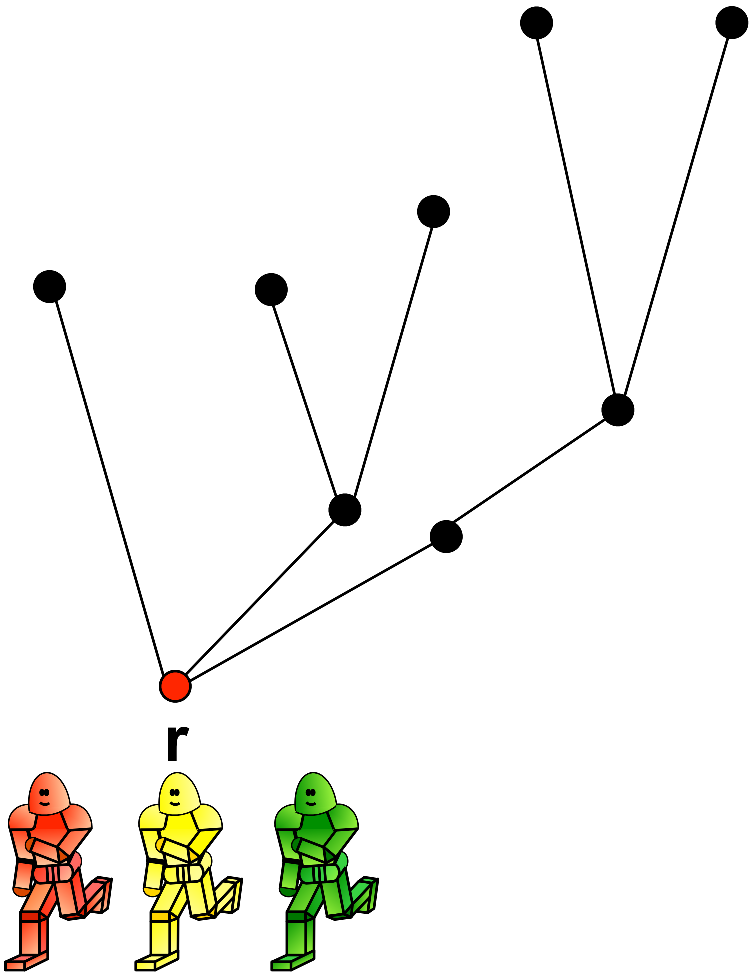
- Lower bounds on actual OPT:
  - Known MAX distance

A New Strategy for General Trees



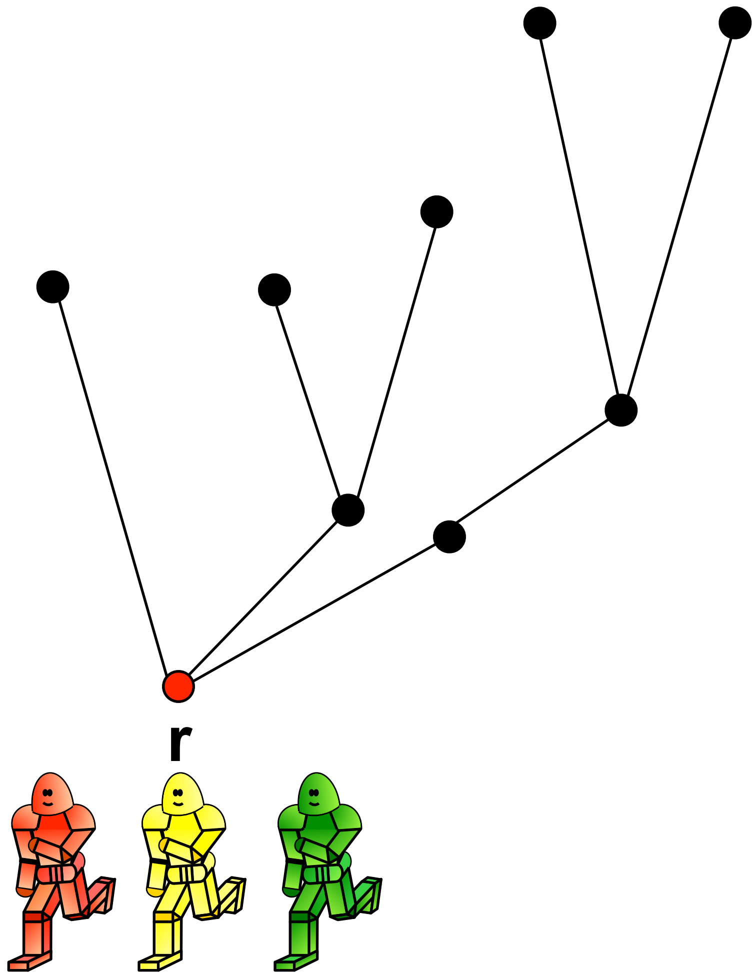
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance

A New Strategy for General Trees



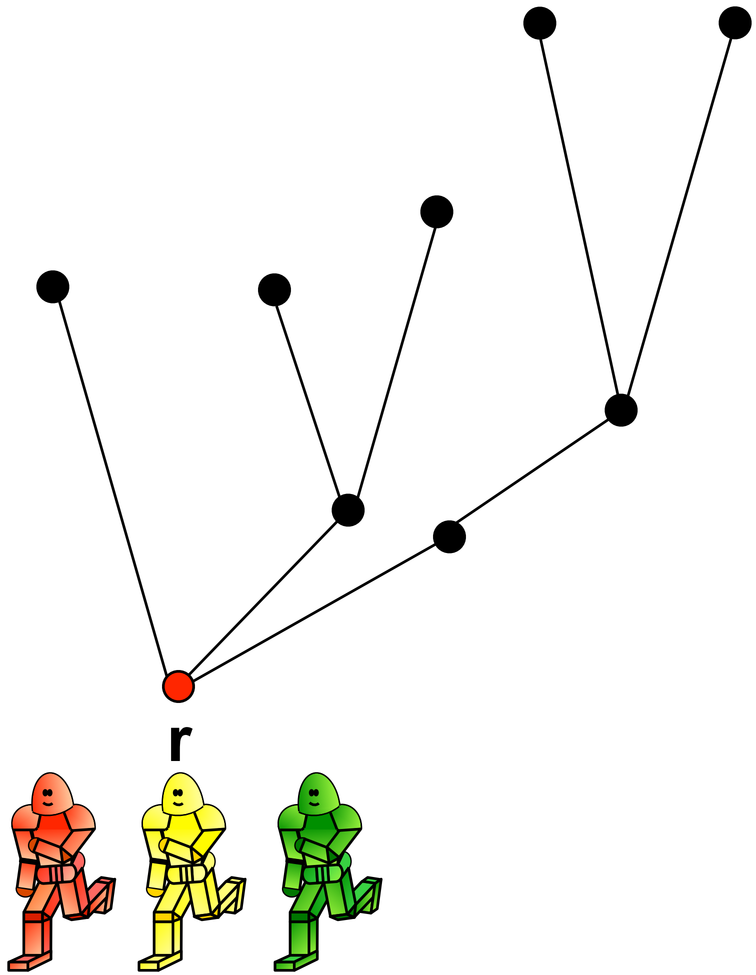
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:

A New Strategy for General Trees



- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.

A New Strategy for General Trees

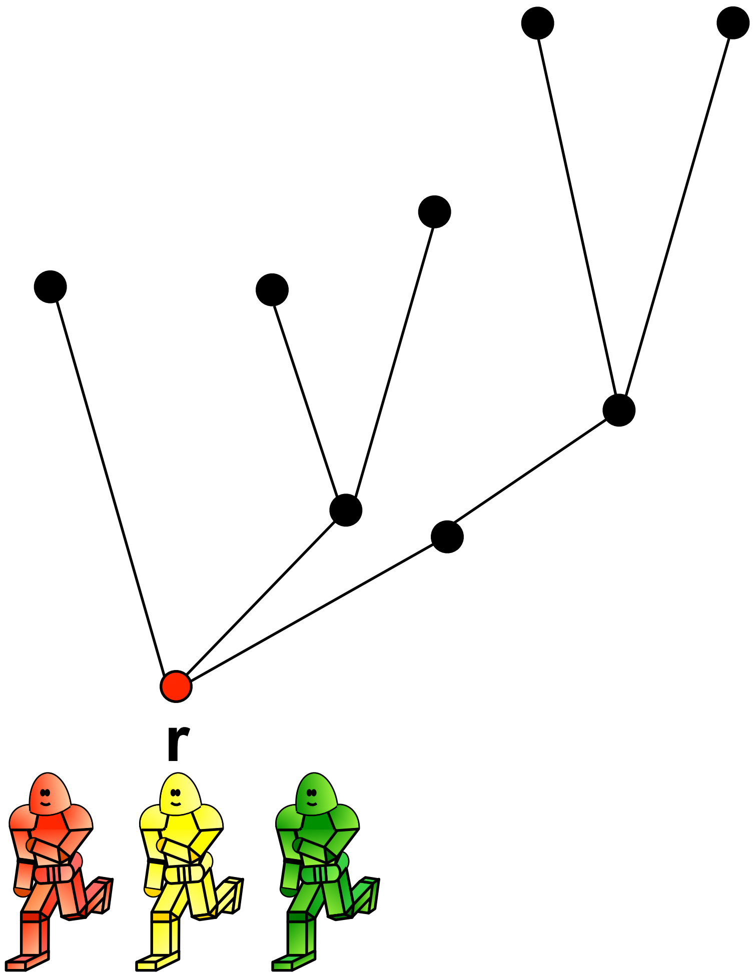


- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.

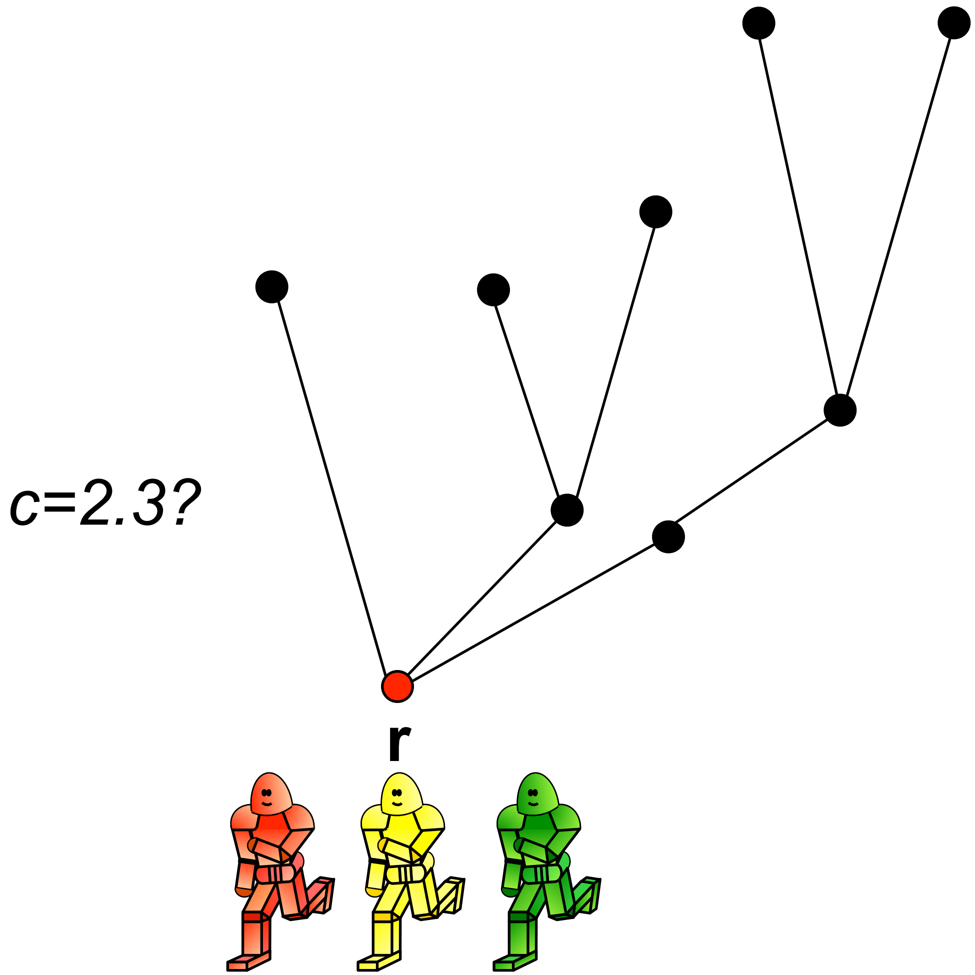
A New Strategy for General Trees

- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.

c=2.3?

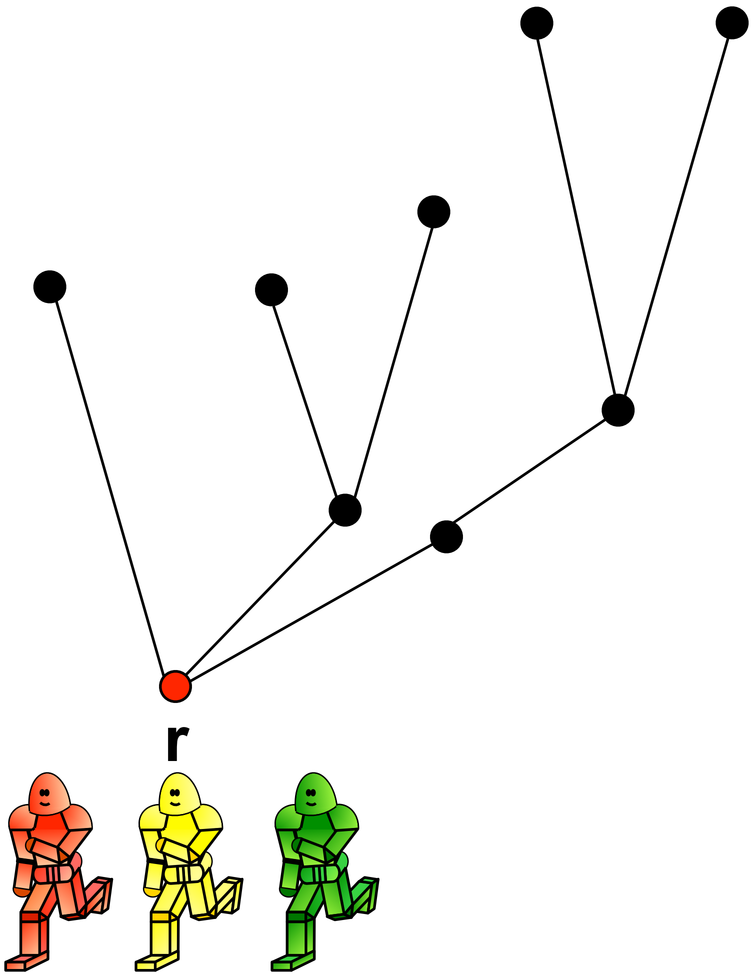


A New Strategy for General Trees



- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.

A New Strategy for General Trees

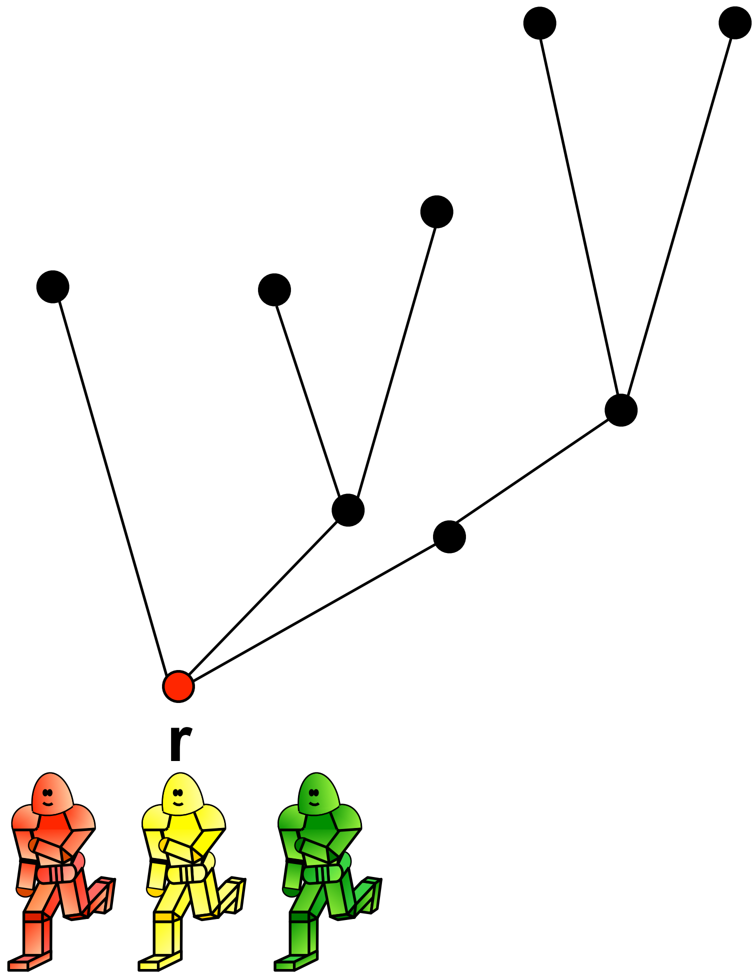


c=2.3?

- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.

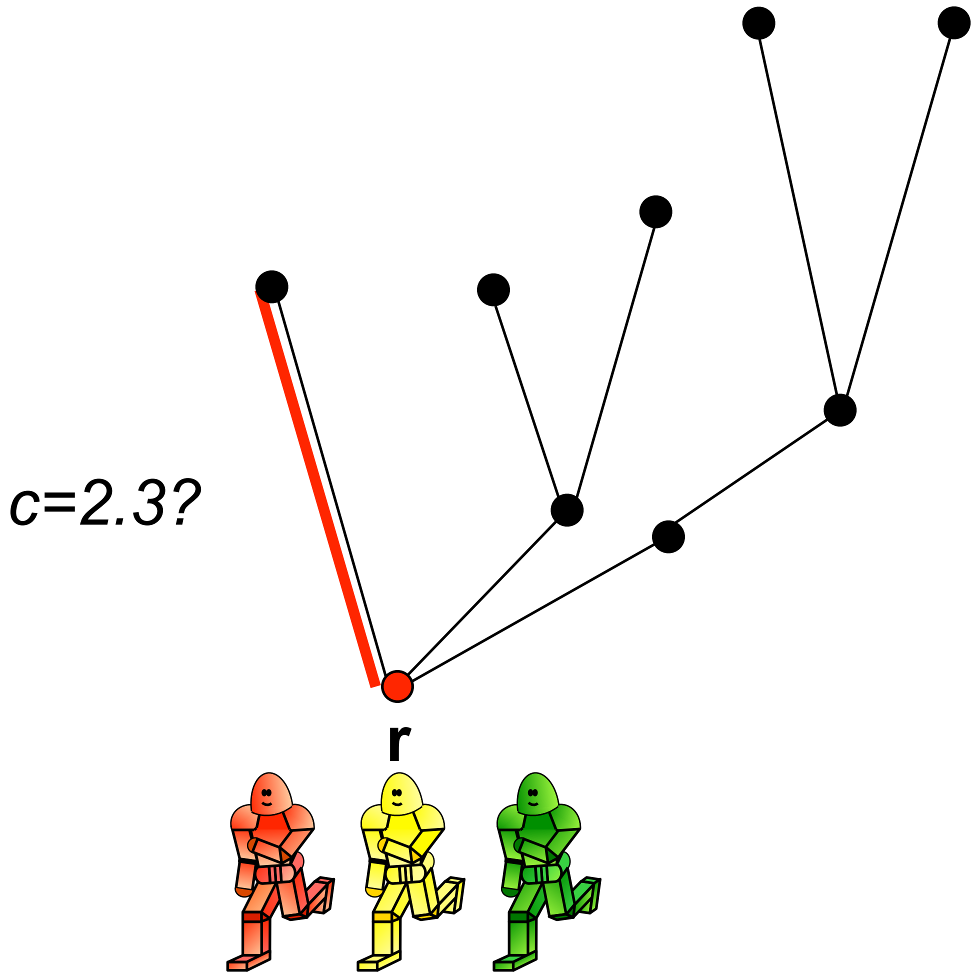
A New Strategy for General Trees

c=2.3?



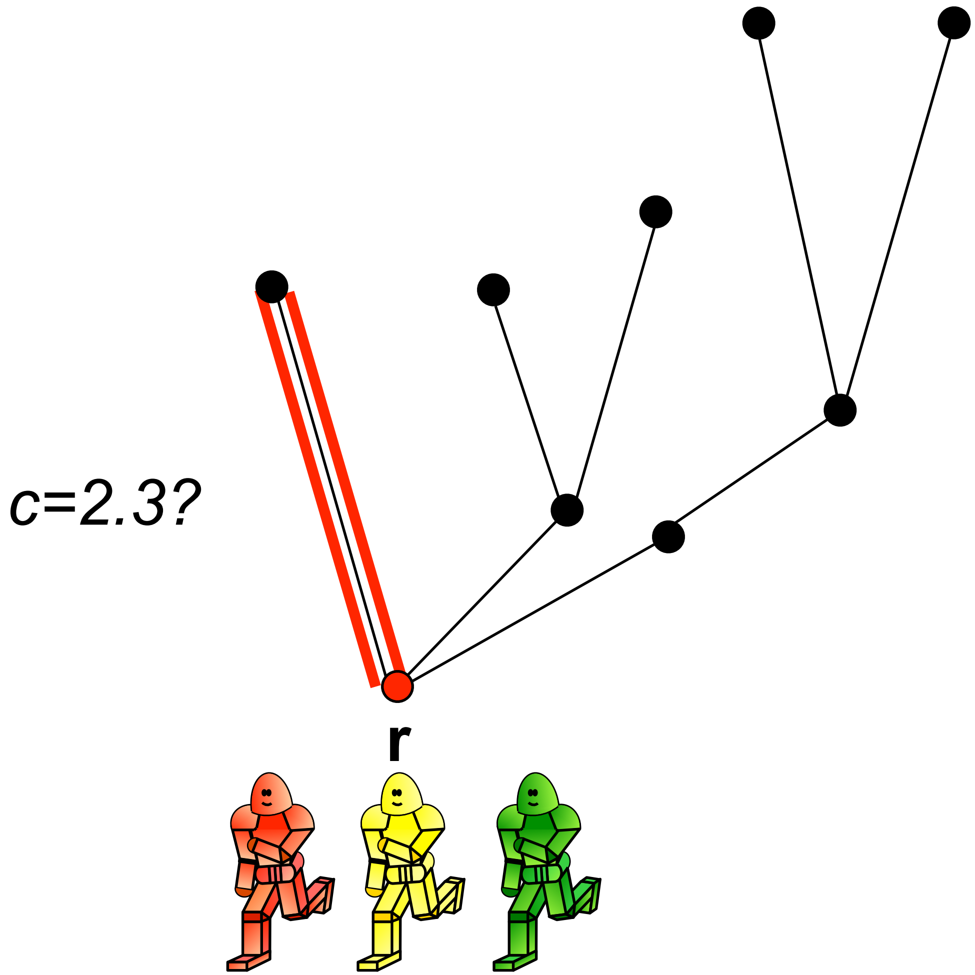
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



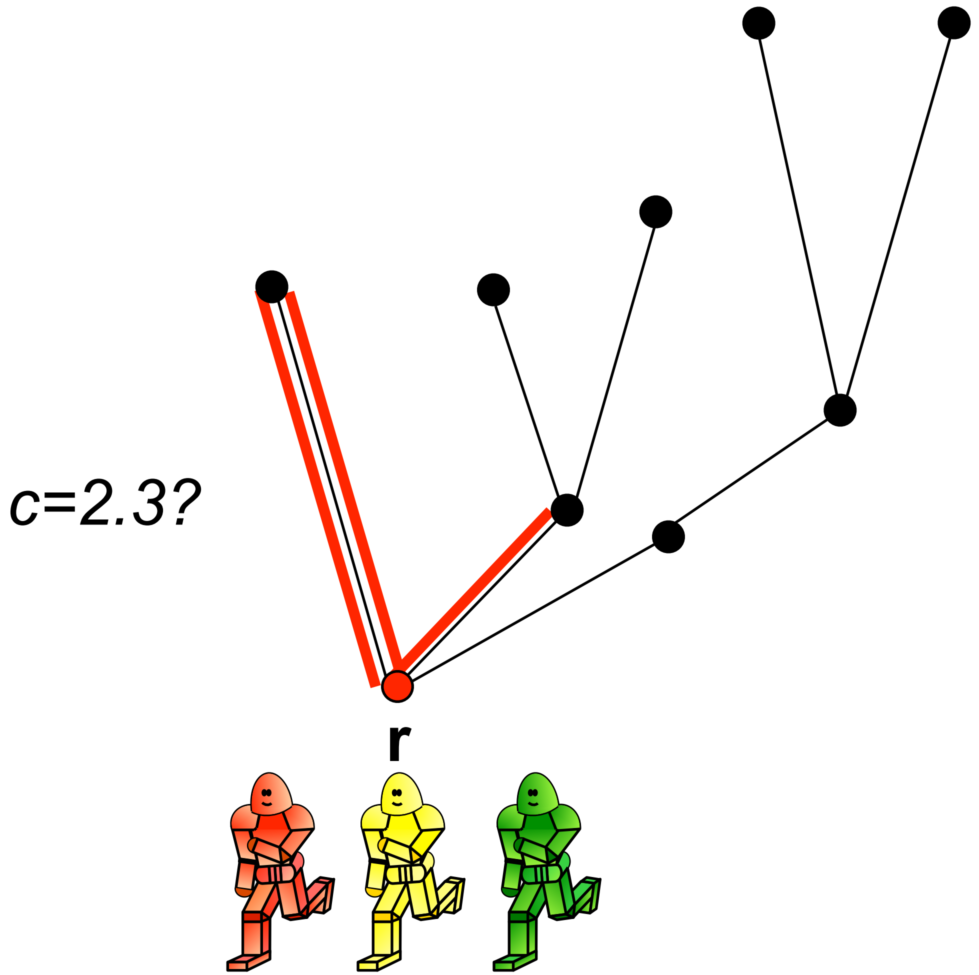
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



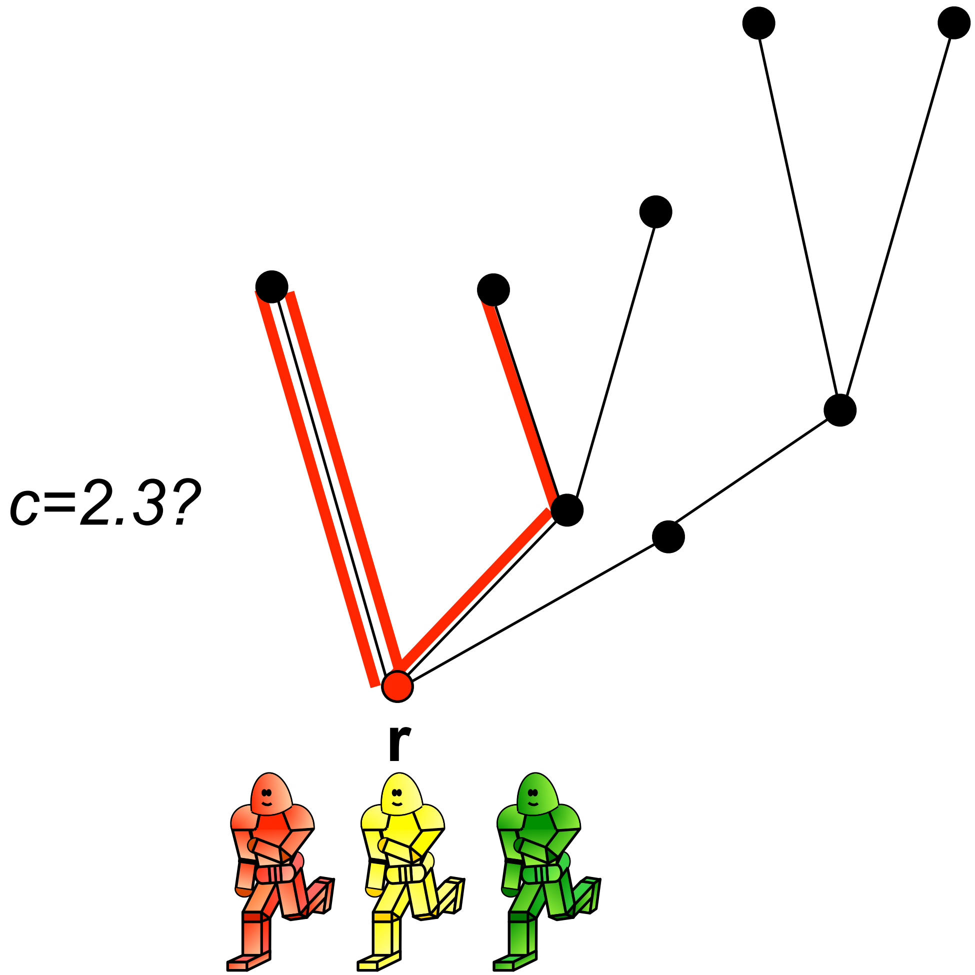
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



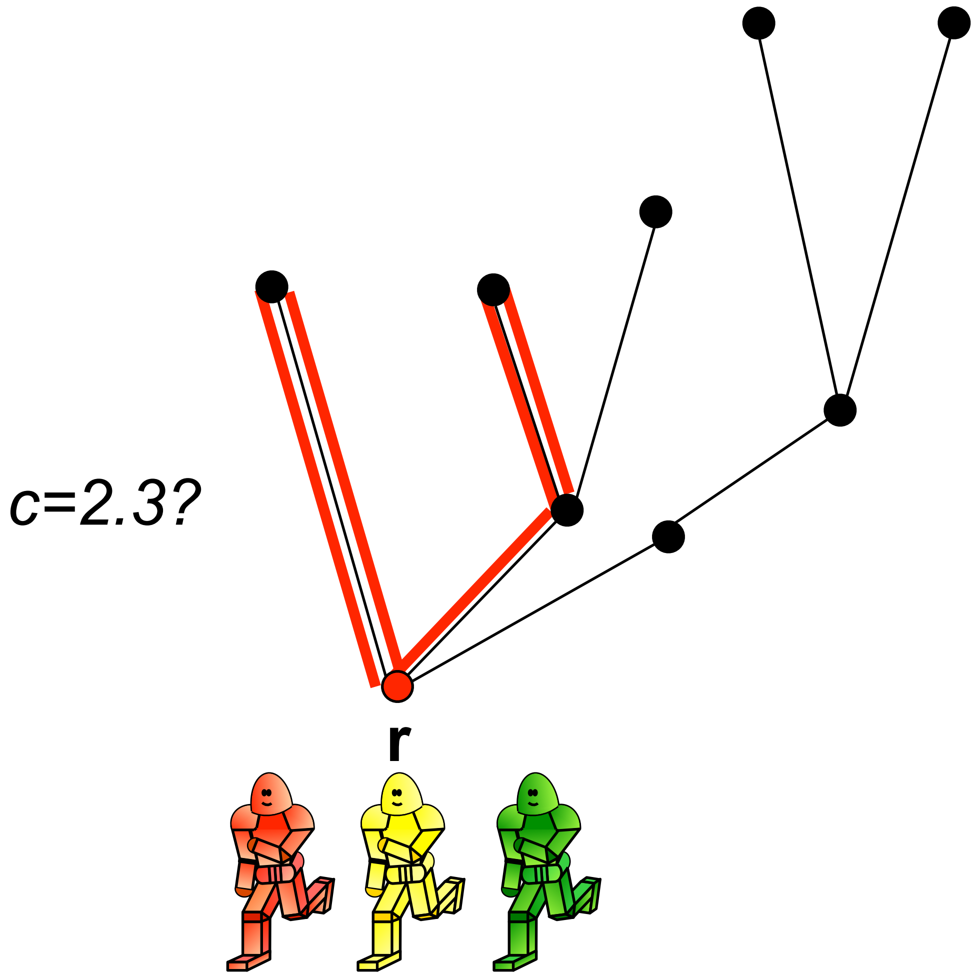
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



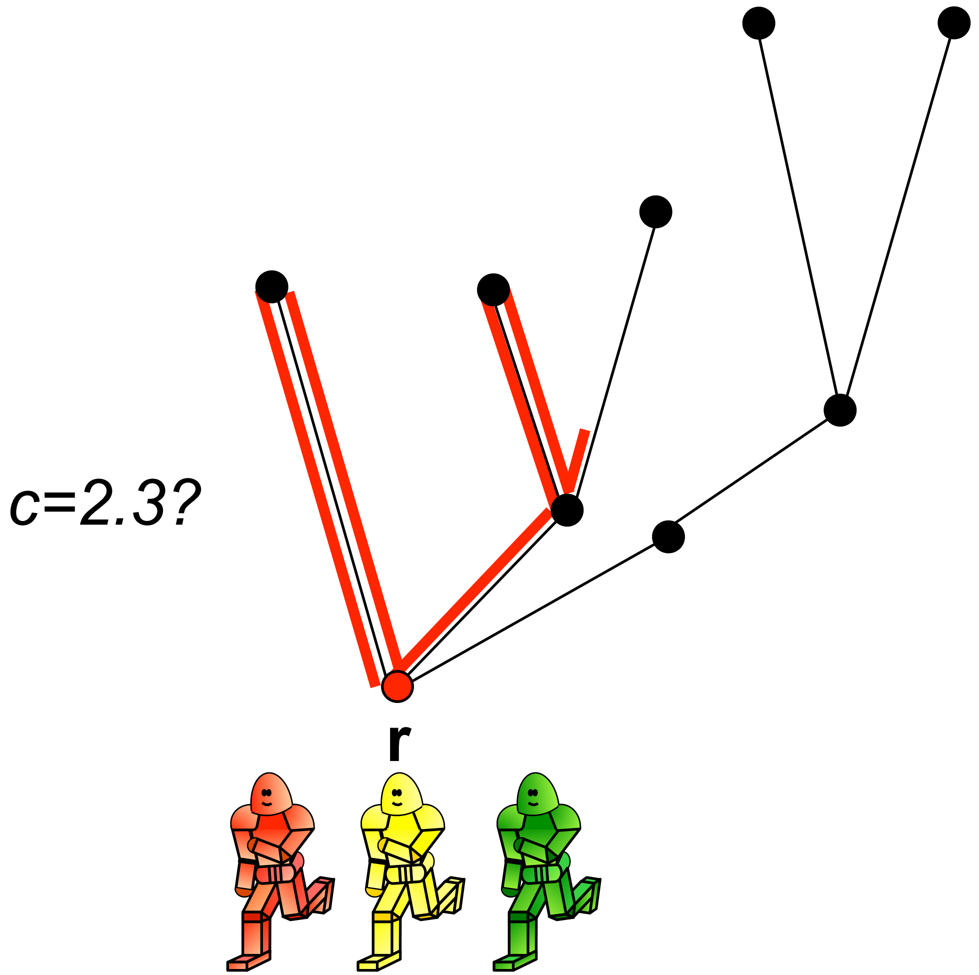
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



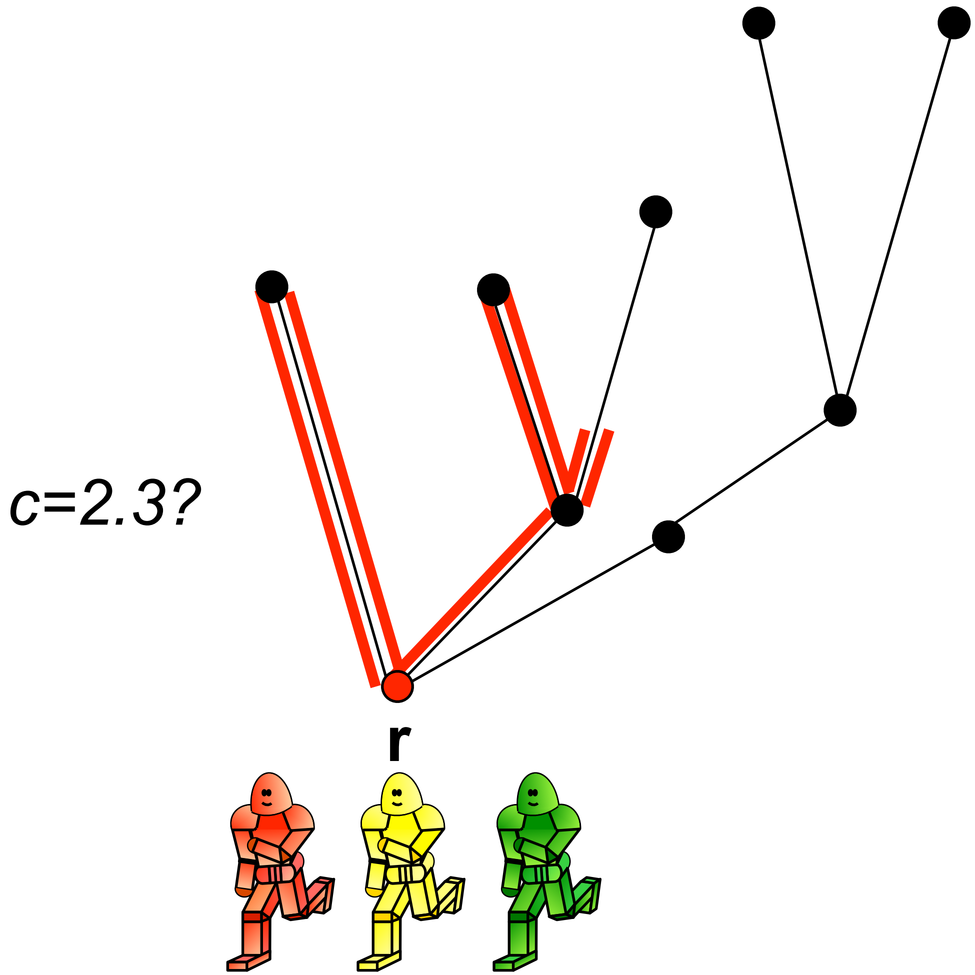
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



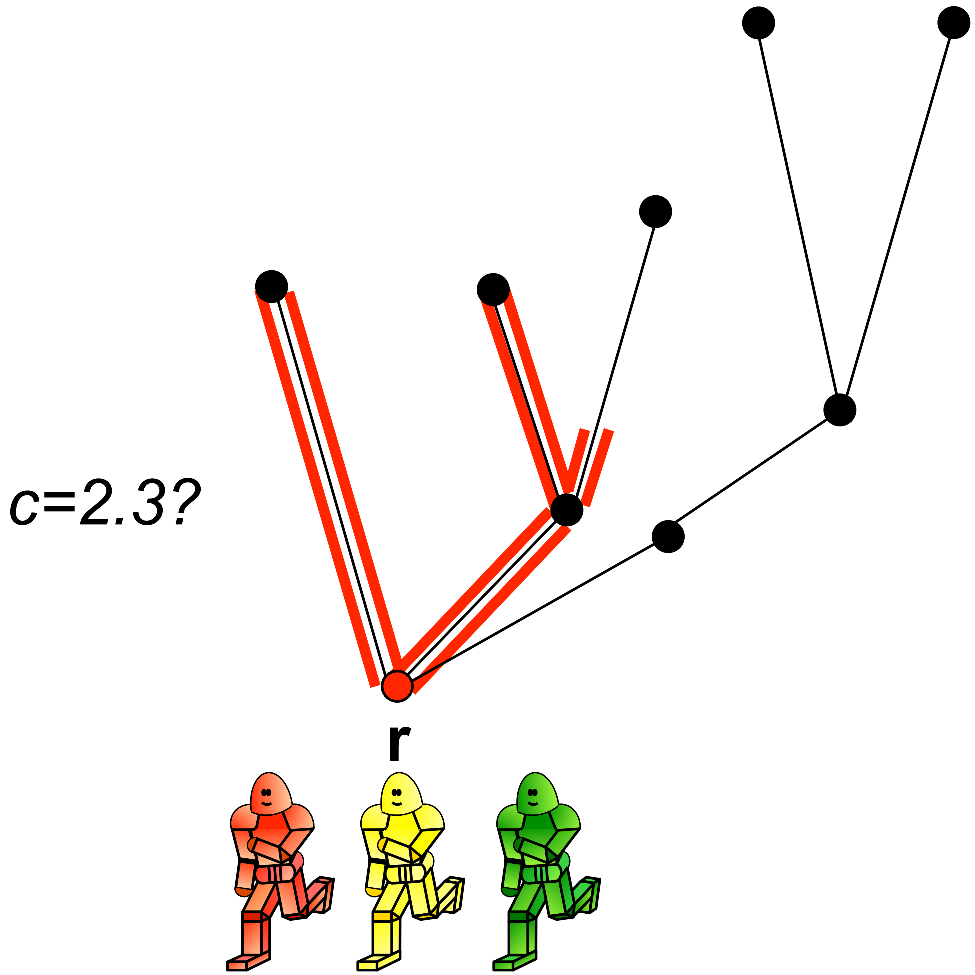
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



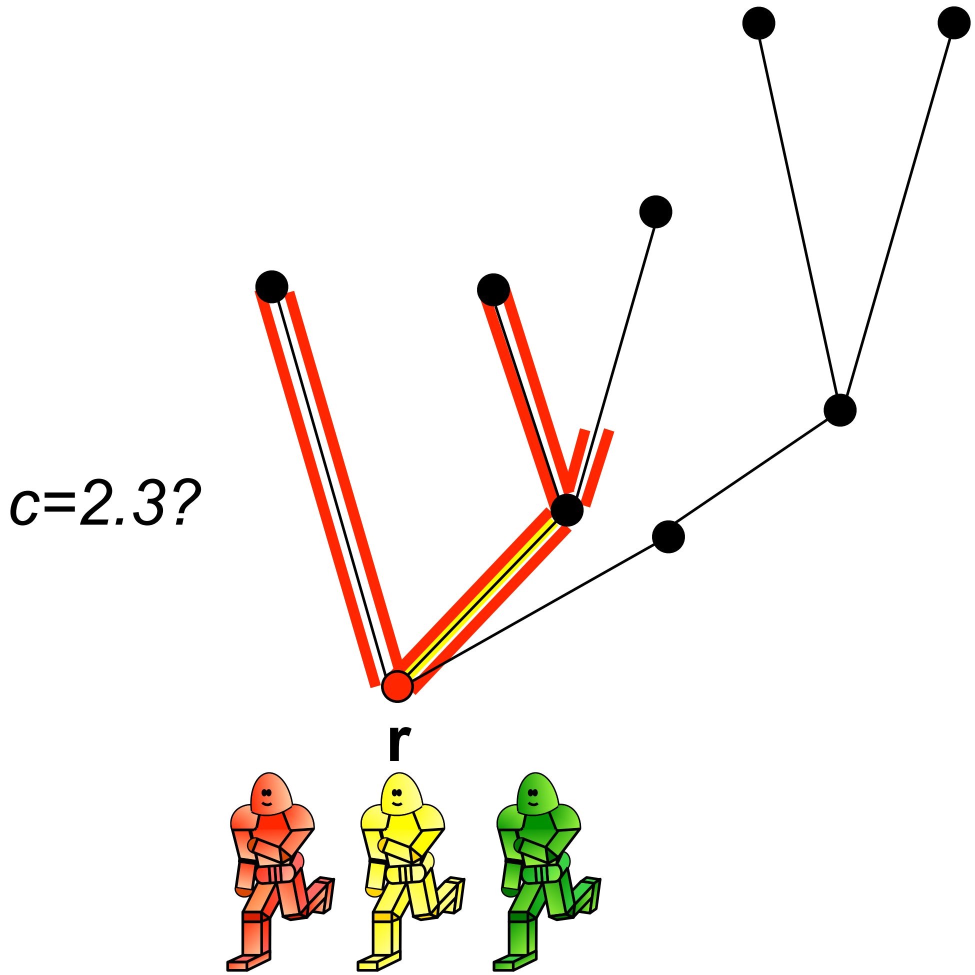
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



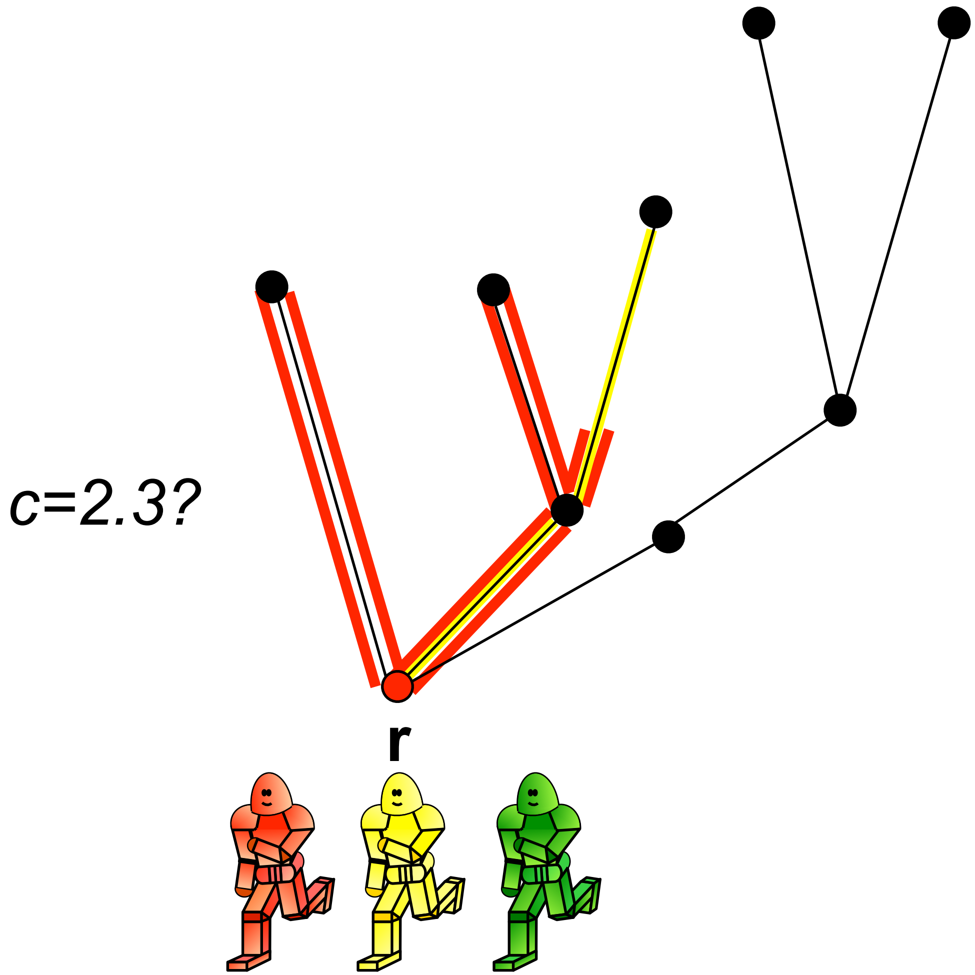
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



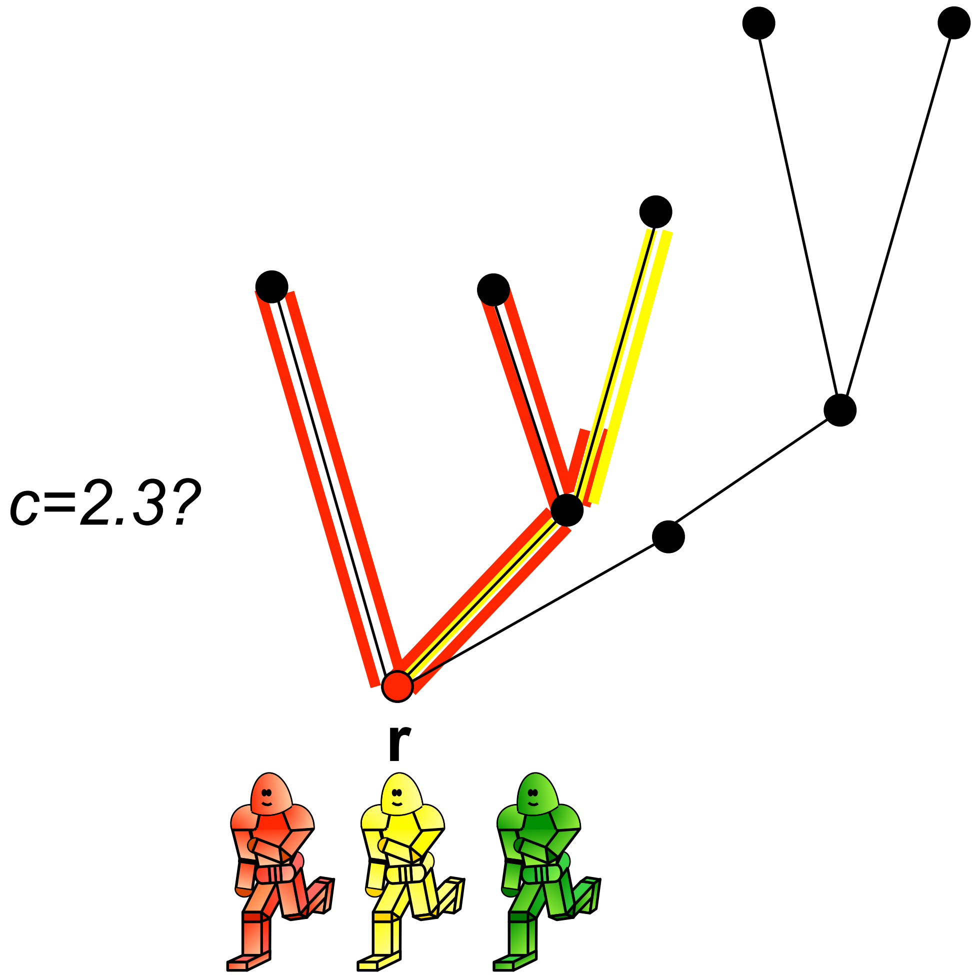
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



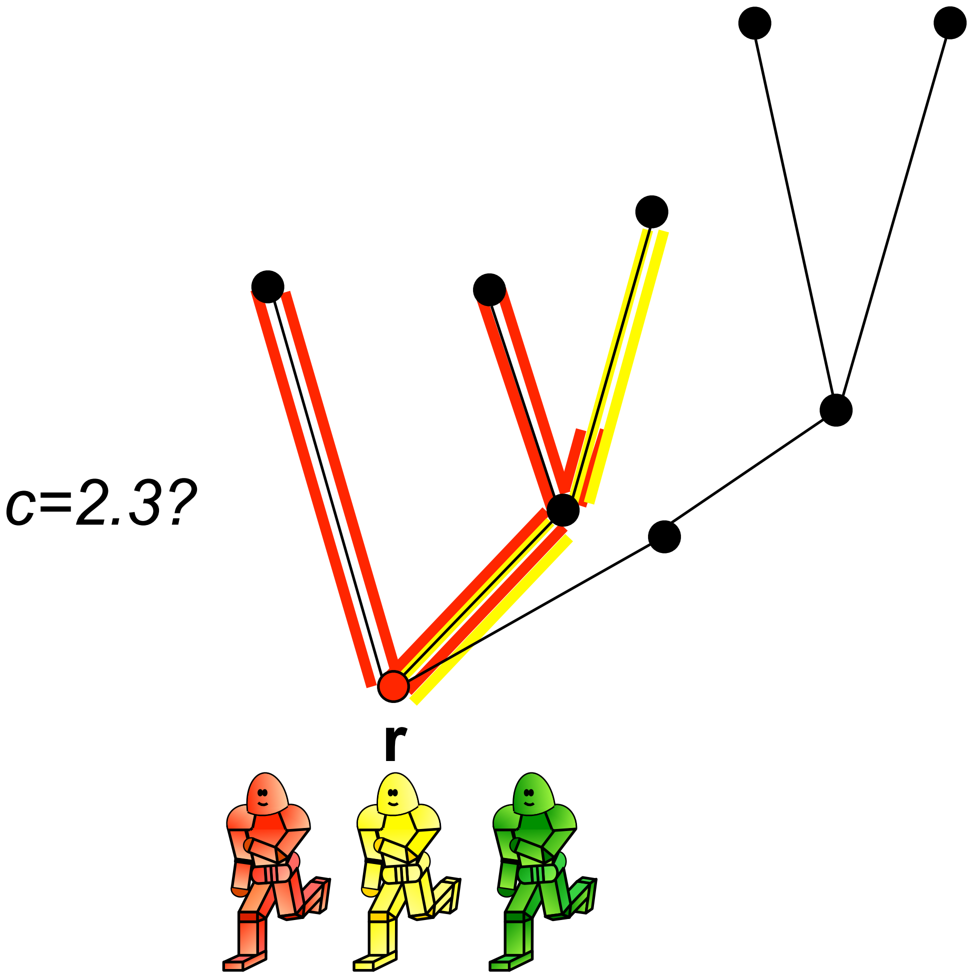
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



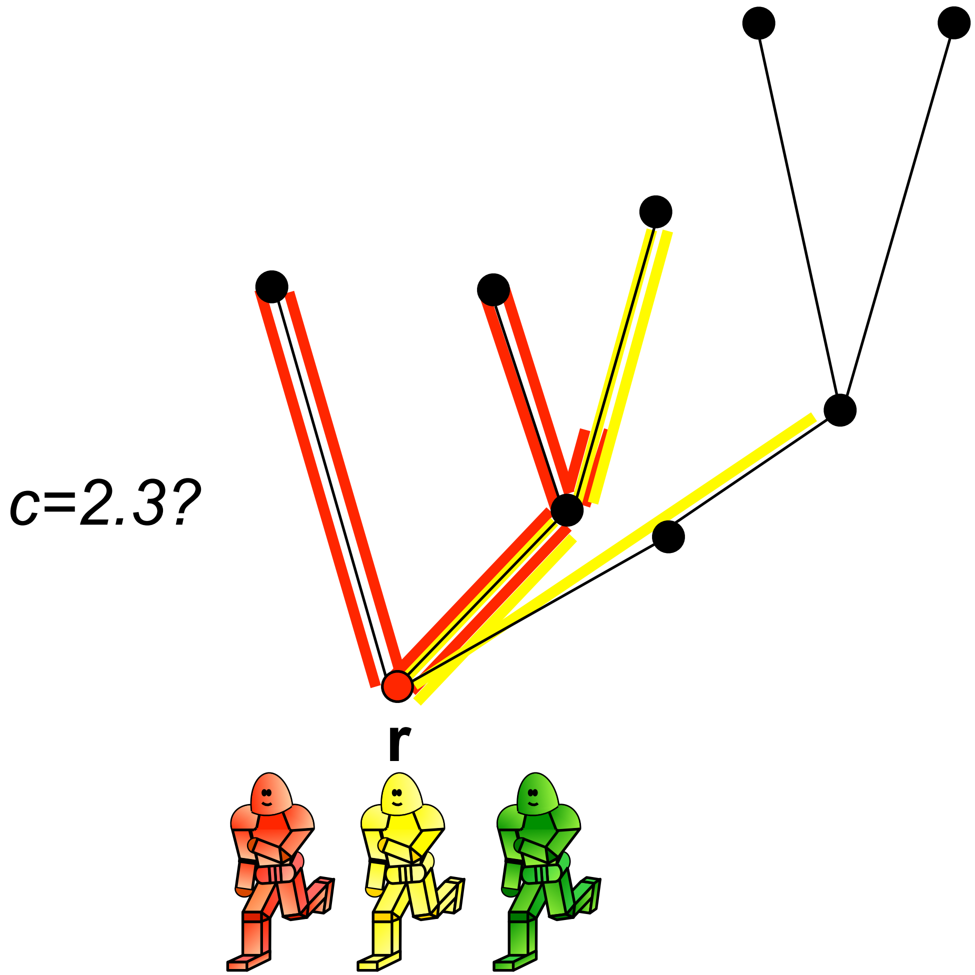
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



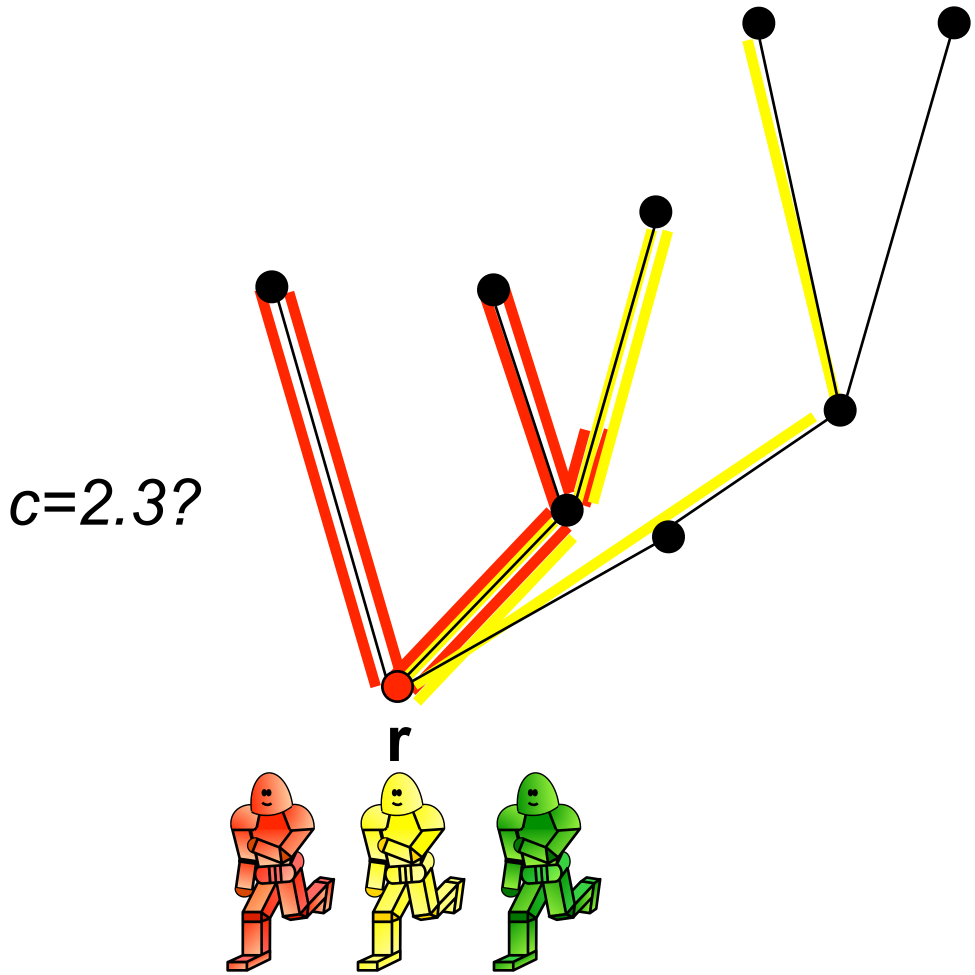
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



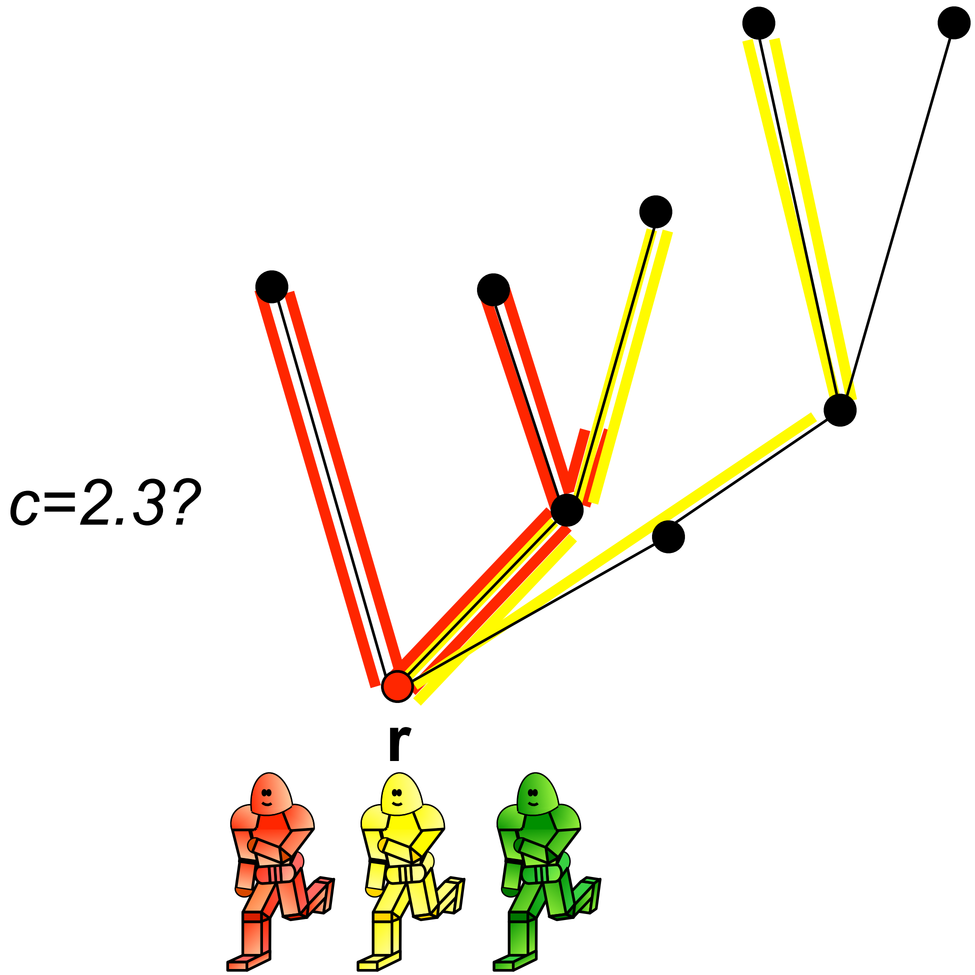
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



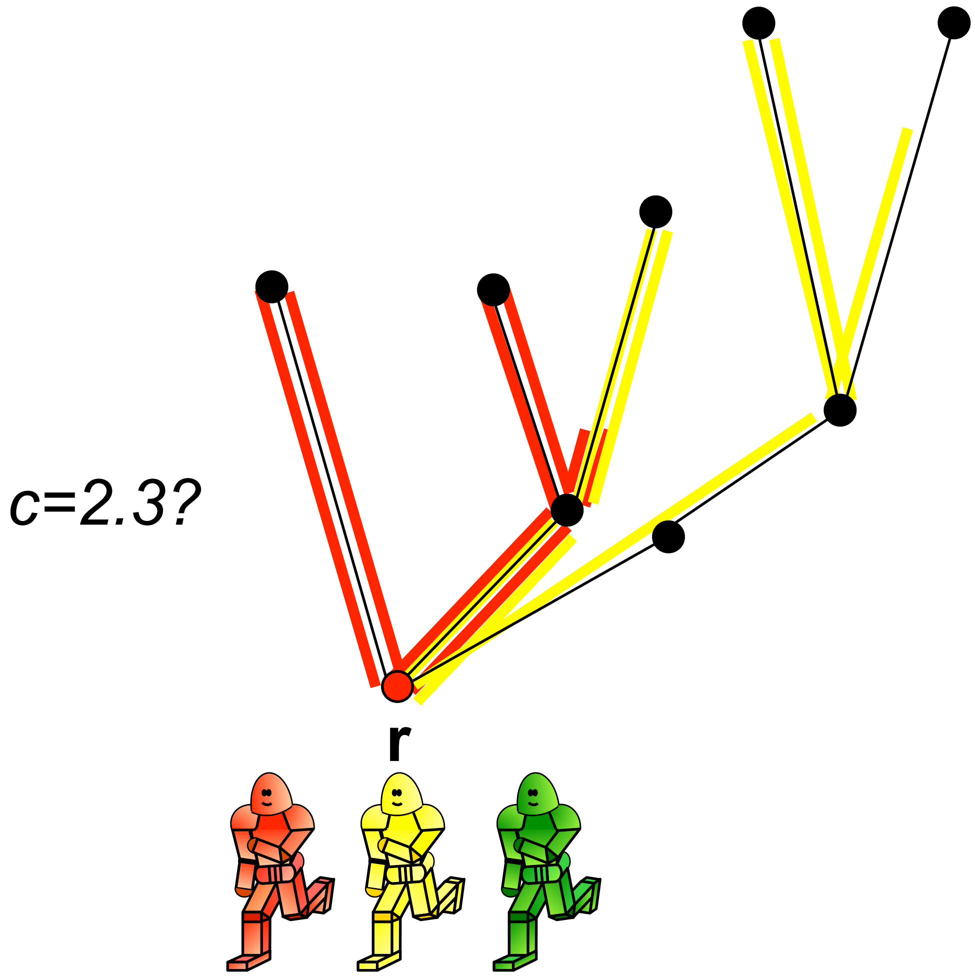
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



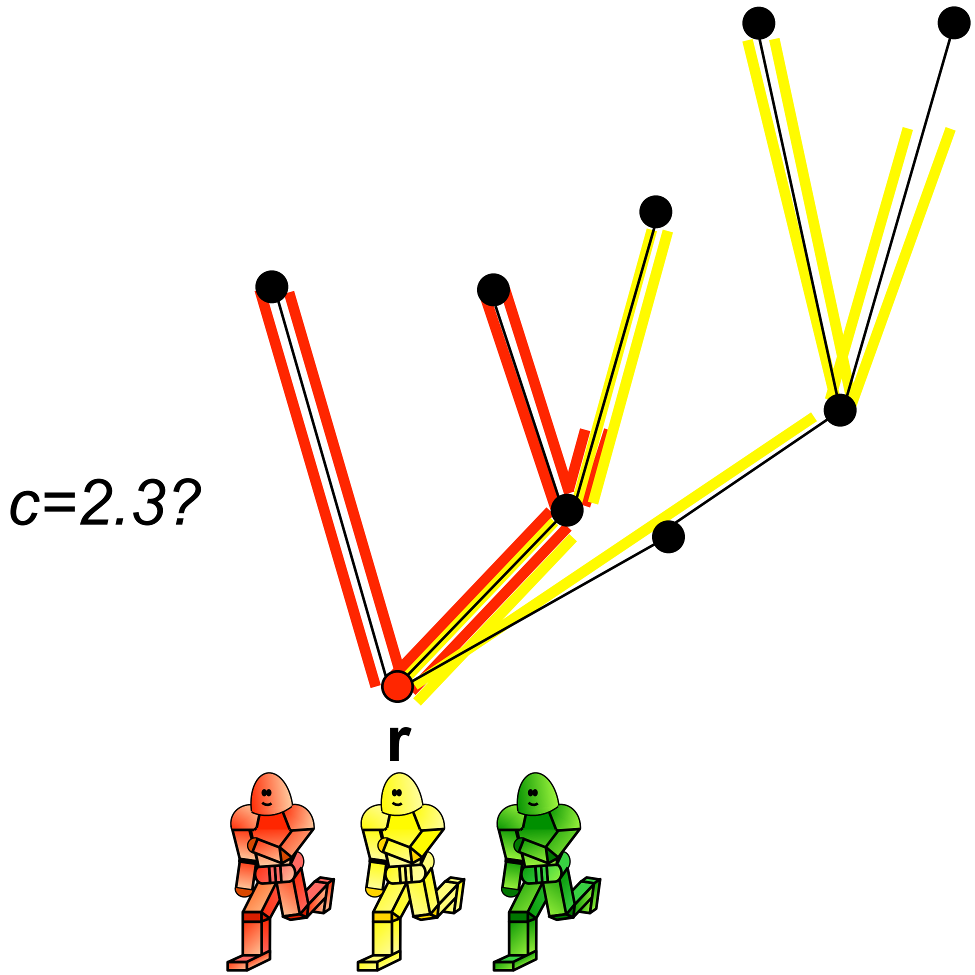
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



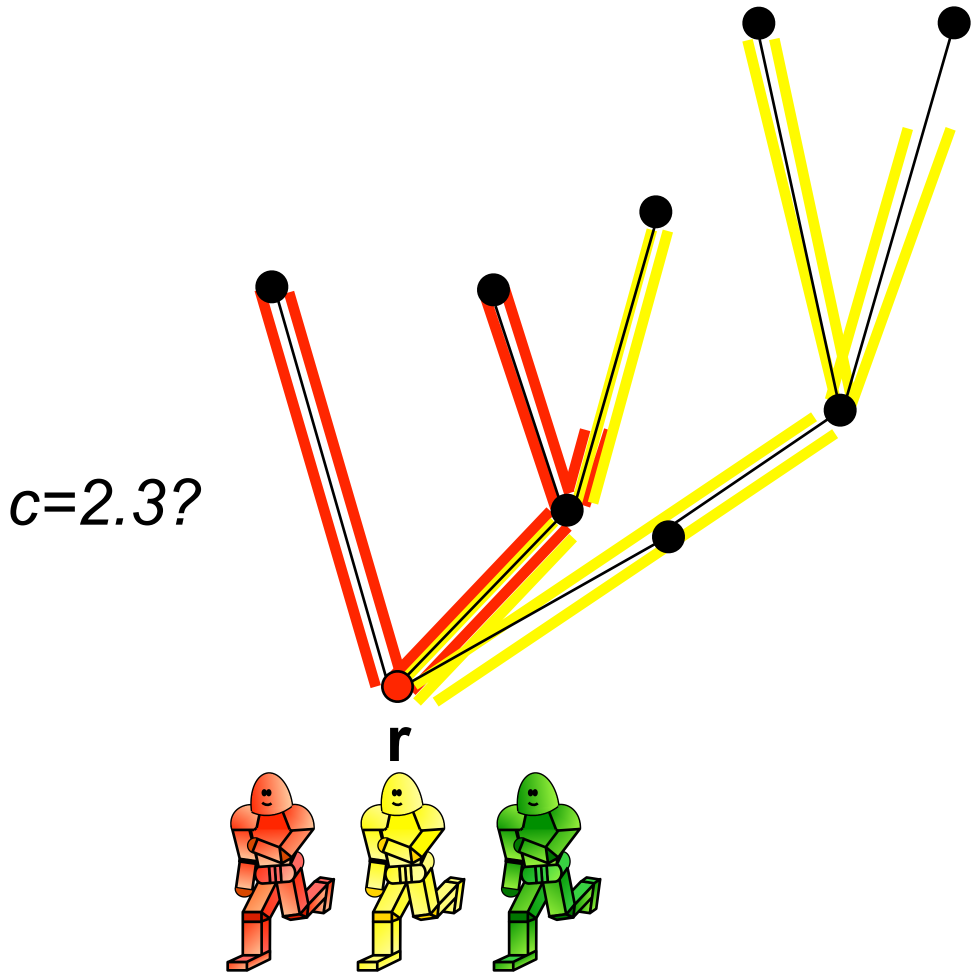
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



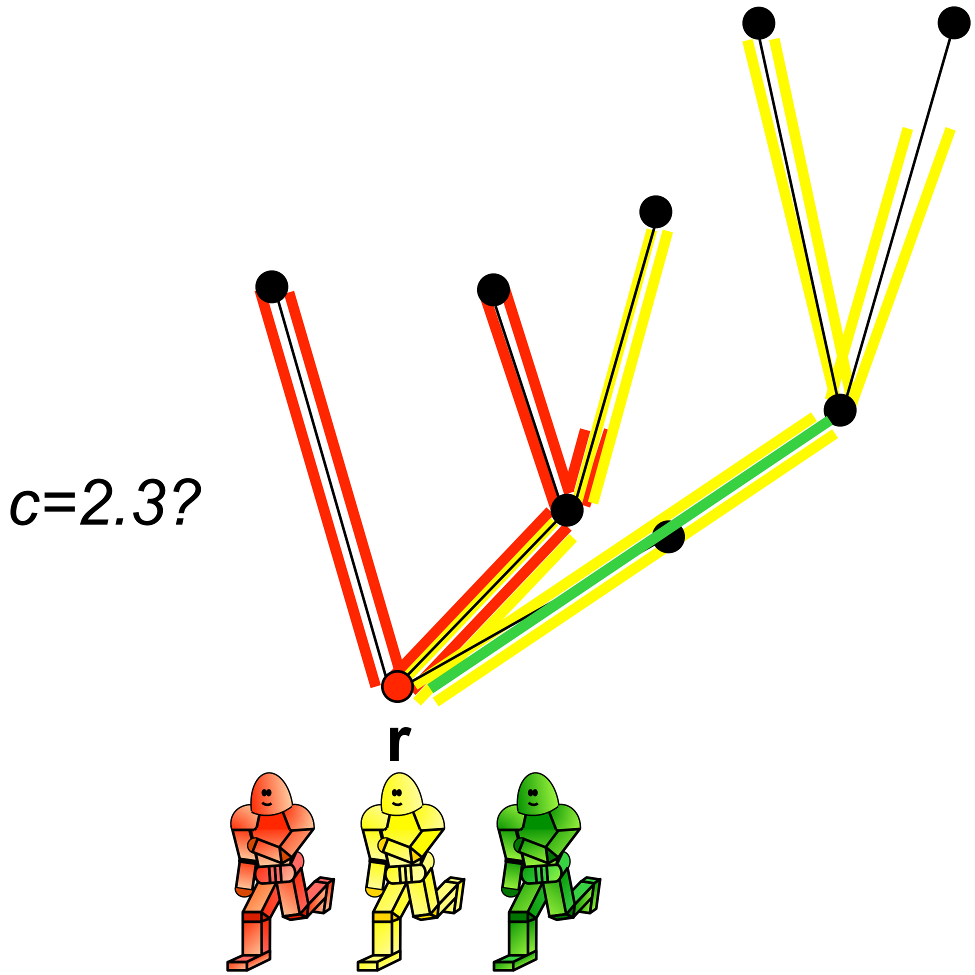
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



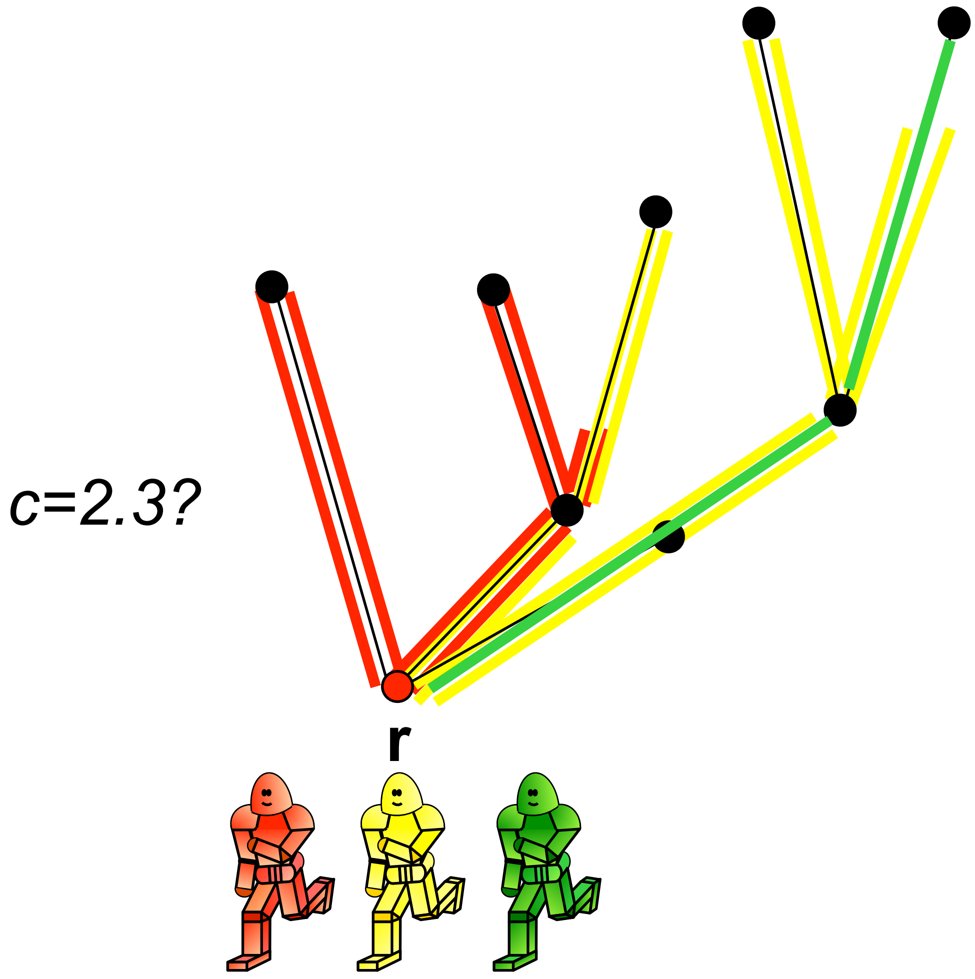
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

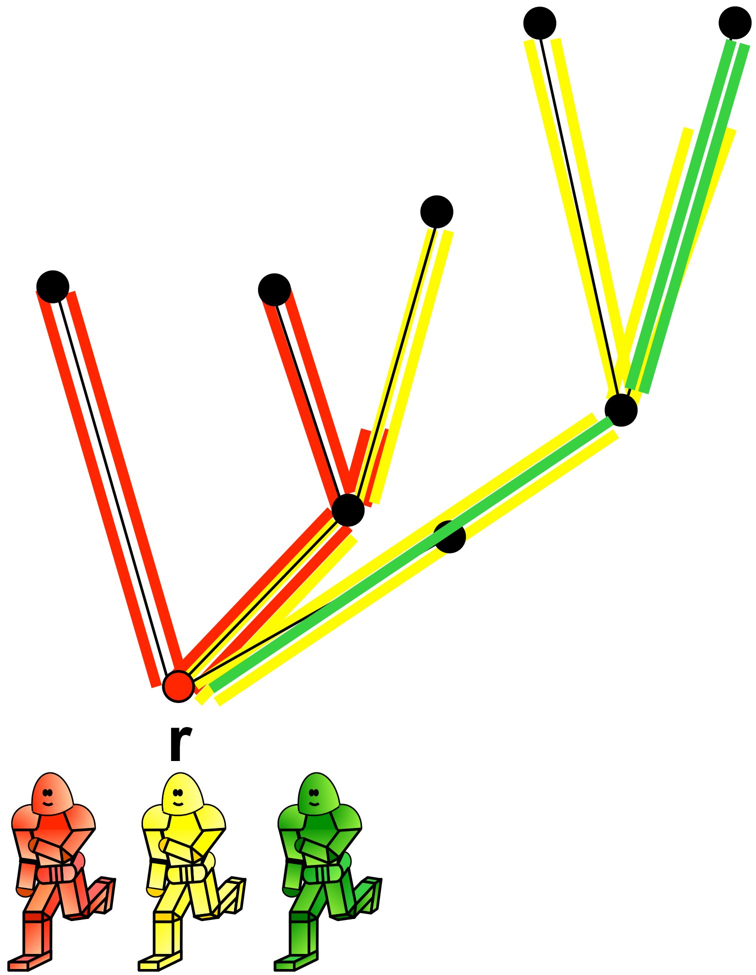
A New Strategy for General Trees



- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

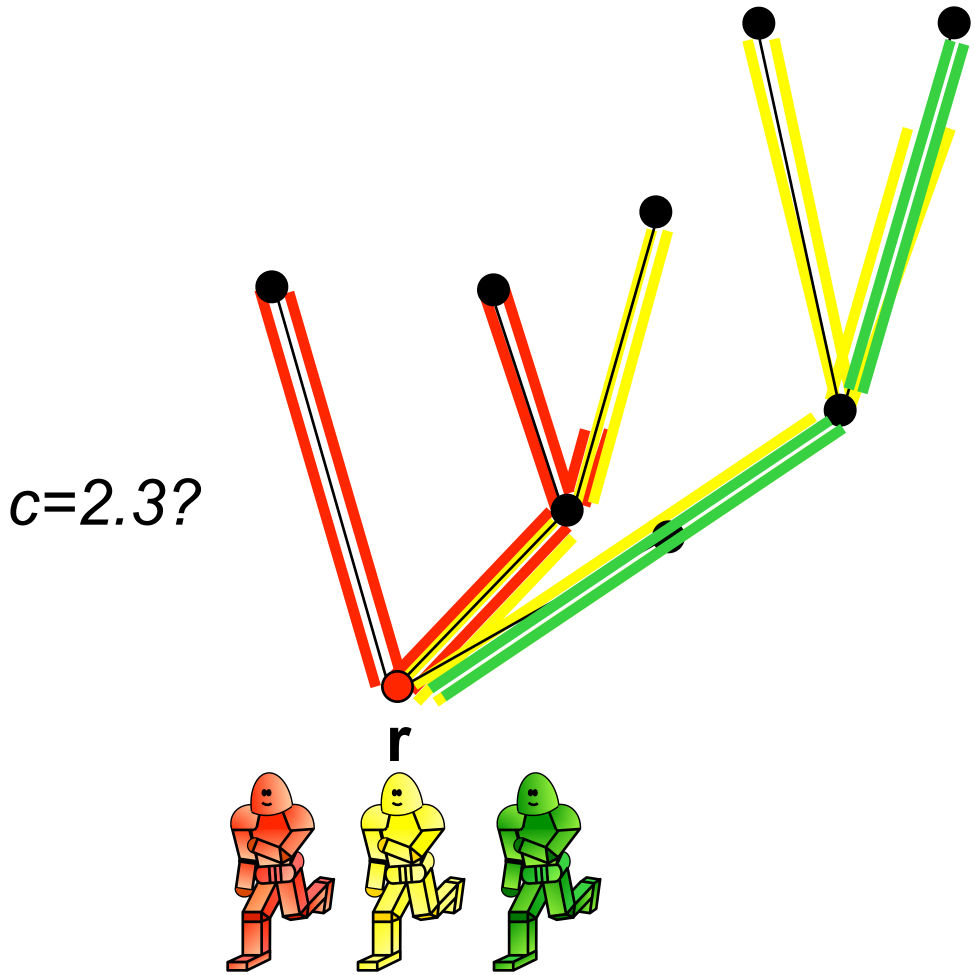
A New Strategy for General Trees

c=2.3?



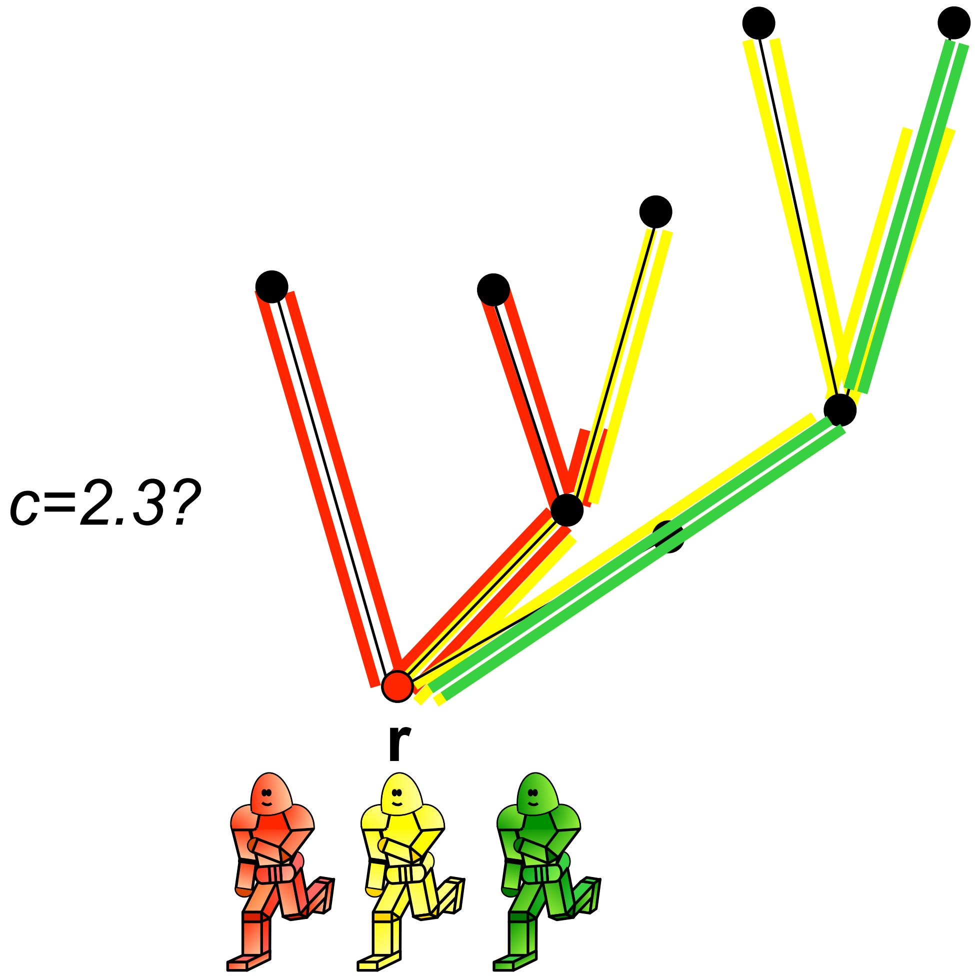
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



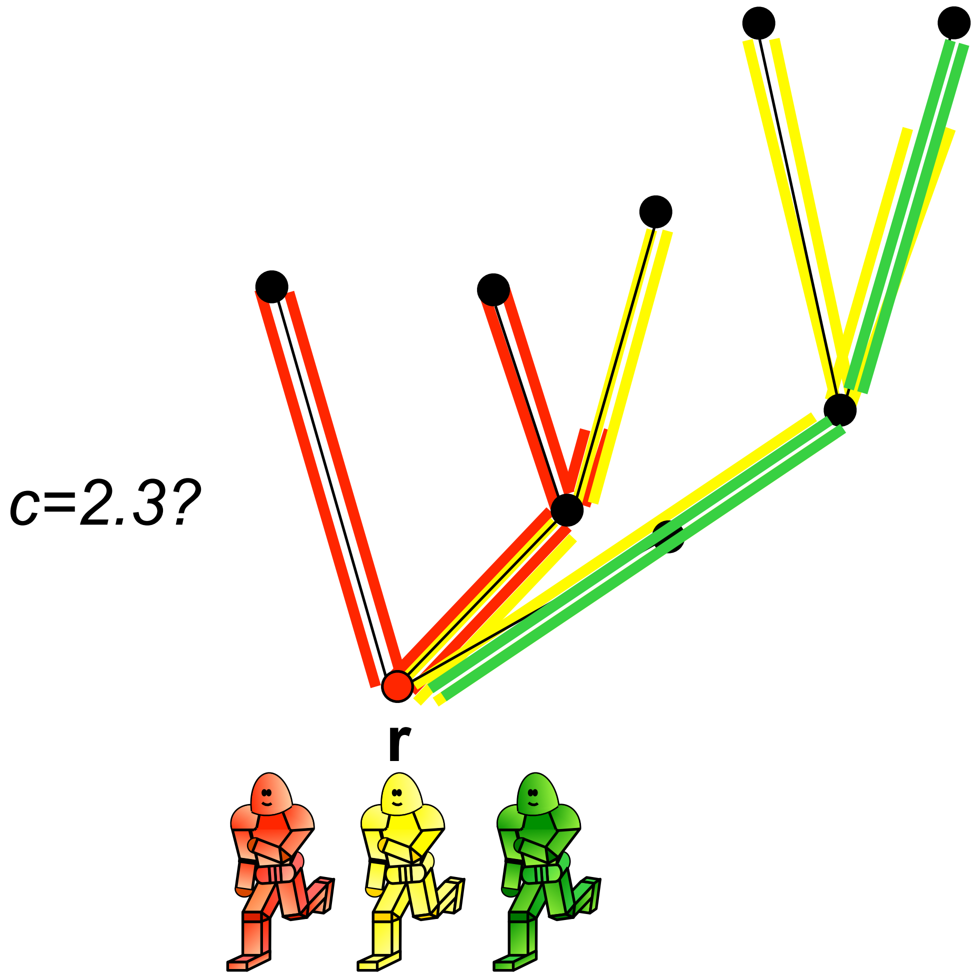
- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.

A New Strategy for General Trees



- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.
- Observations:

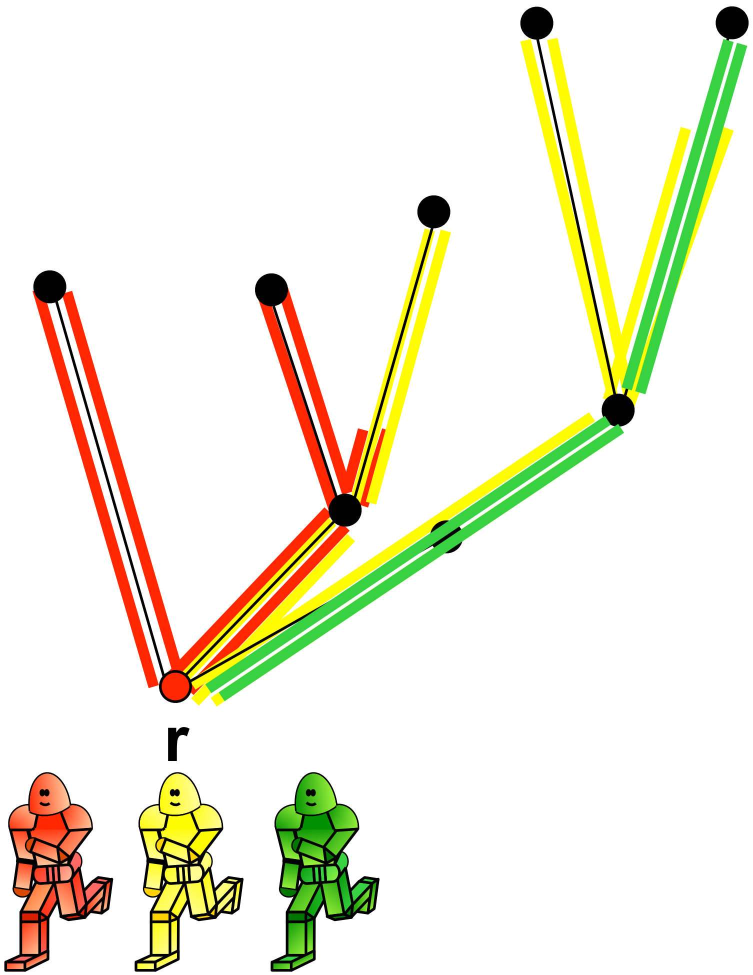
A New Strategy for General Trees



- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.
- Observations:
  - Duplicated distance DUP is bounded by MAX.

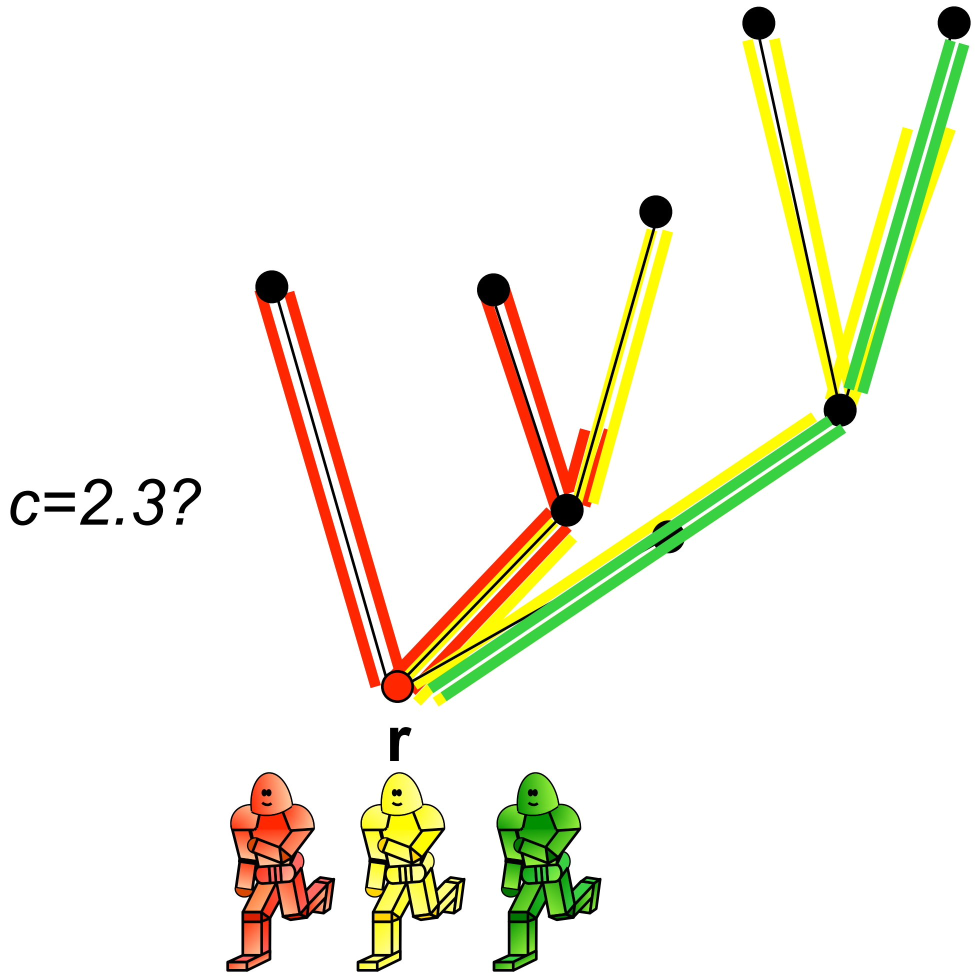
A New Strategy for General Trees

c=2.3?



- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.
- Observations:
  - Duplicated distance DUP is bounded by MAX.
  - In worst case, MAX=AVG=DUP.

A New Strategy for General Trees



- Lower bounds on actual OPT:
  - Known MAX distance
  - AVG of known total distance
- Strategy MAX+AVG:
  - Choose some *c*.
  - Robots take turns, one at a time.
  - Keep track of MAX and AVG.
  - Travel c times lower bound.
- Factor *c* is achievable, if we can keep going - so if we can travel arbitrarily far.
- Observations:
  - Duplicated distance DUP is bounded by MAX.
  - In worst case, MAX=AVG=DUP.
  - This yields a recursion for distances traveled.

A New Strategy for General Trees

A New Strategy for General Trees

Online Balanced Tree Exploration

Pravesh Agrawal
Department of CSE, IIT Bombay, Mumbai, India

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Braunschweig, Germany

Abstract

We study *Online Balanced Tree Exploration*, a class of online optimization problems that can be seen as natural generalizations of both online exploration and machine scheduling: Given an unknown weighted tree $T = (V, E)$ with a distinguished root node r, and a set of $k \geq 2$ identical robots at r, the task is to have all vertices of the tree be visited by some robot and have all robots return to r, such that the largest distance traveled by any robot is minimized. Online Balanced Tree Exploration has been considered before; the best previously known competitive method uses a doubling strategy and yields a factor of 8.

We develop c-GAME, a strategy that proceeds greedily while keeping track of tree depth and average load, and show that it yields a c-competitive strategy for any k and any $c \geq \gamma = 3.146193220582\ldots$, which is tight. Here $\gamma = -W_{-1}(-\frac{1}{e^2})$, where W_{-1} is the lower branch of Lambert's W-function, which is also known as the product logarithm. We also provide a tight characterization of the critical competitive factors γ_k for any specific $k \geq 3$; in particular, we establish $\gamma_3 = 2.27883\ldots$, $\gamma_4 = 2.49221\ldots$, $\gamma_{18} = 2.99961\ldots$, implying that 3-GAME is 3-competitive for all $k \leq 18$.

2012 ACM Subject Classification Theory of computation → Online algorithms; Computing methodologies → Planning and scheduling

Keywords and phrases Online search, group exploration, balanced allocation, competitive analysis

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2022.118

A Useful Lemma

A Useful Lemma

▶ **Lemma 3.** *For analyzing the worst case for strategy c-GAME with* $k > c > 2$*, it suffices to consider*

A Useful Lemma

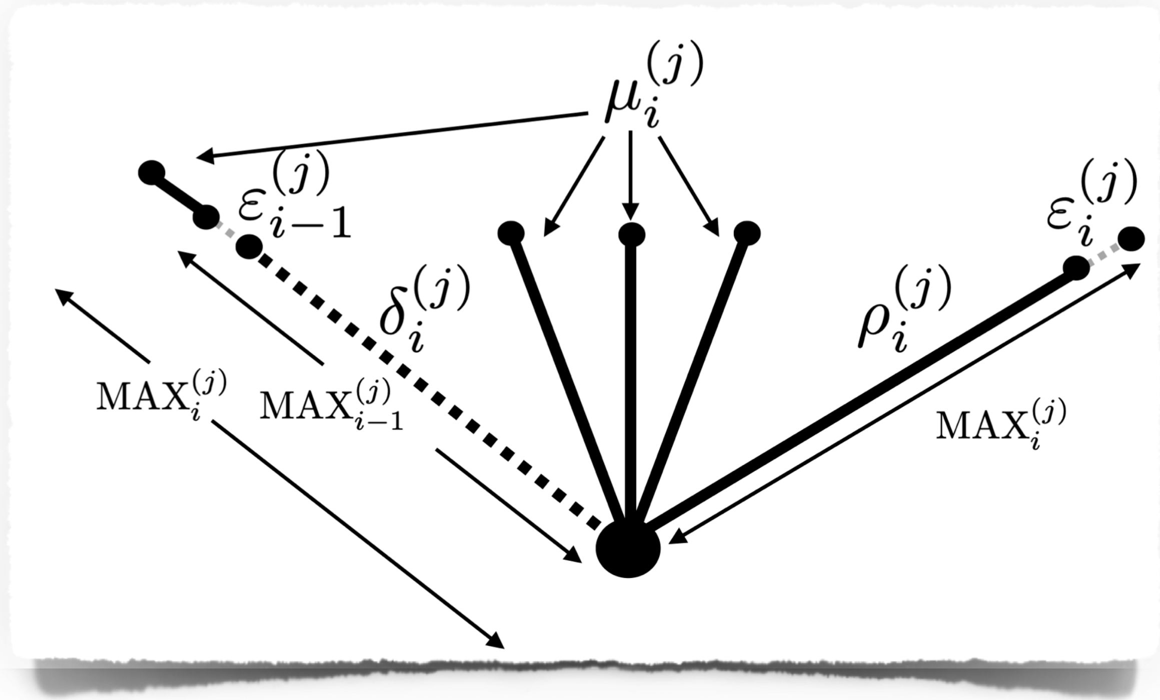
▶ **Lemma 3.** *For analyzing the worst case for strategy c-GAME with $k > c > 2$, it suffices to consider*

1. $\delta_i^{(j)} = MAX_{i-1}^{(j)} - \varepsilon_i^{(j)}$ *with* $\varepsilon_i^{(j)} > 0$ *arbitrarily small for all* i, j *after* $i = 1, j = 1$.
2. $MAX_i^{(j)} = AVG_i^{(j)}$ *for all* i *and all* $j \geq 2$.

A Useful Lemma

▶ **Lemma 3.** *For analyzing the worst case for strategy c-GAME with $k > c > 2$, it suffices to consider*

1. $\delta_i^{(j)} = MAX_{i-1}^{(j)} - \varepsilon_i^{(j)}$ *with* $\varepsilon_i^{(j)} > 0$ *arbitrarily small for all* i, j *after* $i = 1, j = 1$.
2. $MAX_i^{(j)} = AVG_i^{(j)}$ *for all* i *and all* $j \geq 2$.



Recursion

Recursion

D_i : total distance traveled by a robot after iteration i

Recursion

D_i : total distance traveled by a robot after iteration i

d_i : new distance traveled by a robot in iteration i

Recursion

D_i : total distance traveled by a robot after iteration i

d_i : new distance traveled by a robot in iteration i

New

Recursion

D_i : total distance traveled by a robot after iteration i

d_i : new distance traveled by a robot in iteration i

New

$$d_i + D_{i-k} + \frac{D_{i-1}}{c} = c\left(\frac{D_{i-1}}{c} + \frac{d_i}{k}\right)$$

Recursion

D_i : total distance traveled by a robot after iteration i

d_i : new distance traveled by a robot in iteration i

New

$$d_i + \underbrace{D_{i-k}}_{\text{old total}} + \frac{D_{i-1}}{c} = c\left(\frac{D_{i-1}}{c} + \frac{d_i}{k}\right)$$

Recursion

D_i : total distance traveled by a robot after iteration i

d_i : new distance traveled by a robot in iteration i

New

$$\underbrace{d_i + D_{i-k}}_{\text{old total}} + \underbrace{\frac{D_{i-1}}{c}}_{\text{duplicated}} = c\left(\frac{D_{i-1}}{c} + \frac{d_i}{k}\right)$$

old total

duplicated

Recursion

D_i : total distance traveled by a robot after iteration i

d_i : new distance traveled by a robot in iteration i

New

$$\underbrace{d_i}_{\text{new}} + \underbrace{D_{i-k}}_{\text{old total}} + \underbrace{\frac{D_{i-1}}{c}}_{\text{duplicated}} = c\left(\frac{D_{i-1}}{c} + \frac{d_i}{k}\right)$$

Recursion

D_i : total distance traveled by a robot after iteration i

d_i : new distance traveled by a robot in iteration i

New

$$\underbrace{d_i}_{\text{new}} + \underbrace{D_{i-k}}_{\text{old total}} + \underbrace{\frac{D_{i-1}}{c}}_{\text{duplicated}} = c\left(\underbrace{\frac{D_{i-1}}{c}}_{\text{old average}} + \frac{d_i}{k}\right)$$

Recursion

D_i : total distance traveled by a robot after iteration i

d_i : new distance traveled by a robot in iteration i

New

$$\underbrace{d_i}_{\text{new}} + \underbrace{D_{i-k}}_{\text{old total}} + \underbrace{\frac{D_{i-1}}{c}}_{\text{duplicated}} = c\left(\underbrace{\frac{D_{i-1}}{c}}_{\text{old average}} + \underbrace{\frac{d_i}{k}}_{\text{added to average}}\right)$$

Recursion

D_i : total distance traveled by a robot after iteration i

d_i : new distance traveled by a robot in iteration i

New

$$\underbrace{d_i}_{\text{new}} + \underbrace{D_{i-k}}_{\text{old total}} + \underbrace{\frac{D_{i-1}}{c}}_{\text{duplicated}} = c\left(\underbrace{\frac{D_{i-1}}{c}}_{\text{old average}} + \underbrace{\frac{d_i}{k}}_{\text{added to average}}\right)$$

Rearrange

Recursion

D_i : total distance traveled by a robot after iteration i

d_i : new distance traveled by a robot in iteration i

New

$$\underbrace{d_i}_{\text{new}} + \underbrace{D_{i-k}}_{\text{old total}} + \underbrace{\frac{D_{i-1}}{c}}_{\text{duplicated}} = c\left(\underbrace{\frac{D_{i-1}}{c}}_{\text{old average}} + \underbrace{\frac{d_i}{k}}_{\text{added to average}}\right)$$

Rearrange

$$D_i = \left(\frac{k-1}{k-c}\right) D_{i-1} - \left(\frac{c}{k-c}\right) D_{i-k}$$

Analysis

$$x_k^k - \frac{k-1}{(k-c_k)} x_k^{k-1} + \frac{c_k}{k-c_k} = 0$$

$$x_k^{k-1} \geq \frac{c_k}{c_k - 1}$$

$$x_k > 1$$

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2 x + k^2 - 2k + 1$.*

$$x_k^k - \frac{k-1}{(k-c_k)} x_k^{k-1} + \frac{c_k}{k-c_k} = 0$$

$$x_k^{k-1} \geq \frac{c_k}{c_k - 1}$$

$$x_k > 1$$

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2 x + k^2 - 2k + 1$.*

$$x_k^k - \frac{k-1}{(k-c_k)} x_k^{k-1} + \frac{c_k}{k-c_k} = 0$$

$$x_k^{k-1} = \frac{c_k}{c_k - 1}$$

$$x_k > 1$$

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

$$x_k^k - \frac{k-1}{(k-c_k)} x_k^{k-1} + \frac{c_k}{k-c_k} = 0$$

$$x_k^{k-1} = \frac{c_k}{c_k - 1}$$

$$x_k > 1$$

$$\left(1 + \frac{1}{c_k - 1}\right)^{\frac{1}{k-1}} = 1.$$

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2 x + k^2 - 2k + 1$.*

$$x_k^k - \frac{k-1}{(k - c_k)} x_k^{k-1} + \frac{c_k}{k - c_k} = 0$$

$$x_k^{k-1} \geq \frac{c_k}{c_k - 1}$$

$$x_k > 1$$

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

$$x_k^k - \frac{k-1}{(k-c_k)}x_k^{k-1} + \frac{c_k}{k-c_k} = 0$$

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k - (k-1)x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

$$x_k^k - \frac{k-1}{(k-c_k)} x_k^{k-1} + \frac{c_k}{k-c_k} = 0$$

$$(k-c_k)x_k^k - (k-1)x_k^{k-1} + c_k = 0$$

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k - (k-1)x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

$$x_k^k - \frac{k-1}{(k-c_k)} x_k^{k-1} + \frac{c_k}{k-c_k} = 0$$

$$(k-c_k)x_k^k - (k-1)x_k^{k-1} + c_k = 0$$

$$c_k = \frac{kx_k^k - (k-1)x_k^{k-1}}{x_k^k - 1}.$$

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

$$x_k^k - \frac{k-1}{(k-c_k)} x_k^{k-1} + \frac{c_k}{k-c_k} = 0$$

$$(k-c_k)x_k^k - (k-1)x_k^{k-1} + c_k = 0$$

$$c_k = \frac{kx_k^k - (k-1)x_k^{k-1}}{x_k^k - 1}.$$

Derivative

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies* $c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$, *with $x_k > 1$ being a zero of the function* $f(x) = (k-1)x^k - k^2 x + k^2 - 2k + 1$.

$$x_k^k - \frac{k-1}{(k - c_k)} x_k^{k-1} + \frac{c_k}{k - c_k} = 0$$

$$(k - c_k) x_k^k - (k-1) x_k^{k-1} + c_k = 0$$

$$c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$$

Derivative

$$\frac{x_k^{k-2}\left((k-1)x_k^k - k^2 x_k + k^2 - 2k + 1\right)}{\left(x_k^k - 1\right)^2}$$

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k - (k-1)x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	
3	
4	
5	
6	
7	
8	
9	
10	
20	
40	
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	
4	
5	
6	
7	
8	
9	
10	
20	
40	
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies* $c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$, *with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2 x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	
5	
6	
7	
8	
9	
10	
20	
40	
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	
6	
7	
8	
9	
10	
20	
40	
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k - (k-1)x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	
7	
8	
9	
10	
20	
40	
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k - (k-1)x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	
8	
9	
10	
20	
40	
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2 x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	
9	
10	
20	
40	
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k - (k-1)x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	
10	
20	
40	
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2 x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	2.85363...
10	
20	
40	
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2 x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	2.85363...
10	2.88277...
20	
40	
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k - (k-1)x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	2.85363...
10	2.88277...
20	3.01425...
40	
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	2.85363...
10	2.88277...
20	3.01425...
40	3.08016...
100	
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k - (k-1)x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	2.85363...
10	2.88277...
20	3.01425...
40	3.08016...
100	3.11977...
1,000	
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	2.85363...
10	2.88277...
20	3.01425...
40	3.08016...
100	3.11977...
1,000	3.14
10,000	
100,000	
1,000,000	

Technische Universität Braunschweig

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	2.85363...
10	2.88277...
20	3.01425...
40	3.08016...
100	3.11977...
1,000	3.14355...
10,000	
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k - (k-1)x_k^{k-1}}{x_k^k - 1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	2.85363...
10	2.88277...
20	3.01425...
40	3.08016...
100	3.11977...
1,000	3.14355...
10,000	3.14592...
100,000	
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	2.85363...
10	2.88277...
20	3.01425...
40	3.08016...
100	3.11977...
1,000	3.14355...
10,000	3.14592...
100,000	3.14612...
1,000,000	

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, with $x_k > 1$ being a zero of the function $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.*

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	2.85363...
10	2.88277...
20	3.01425...
40	3.08016...
100	3.11977...
1,000	3.14355...
10,000	3.14592...
100,000	3.14612...
1,000,000	3.14619...

Analysis

▶ **Lemma 1.** *For any fixed k, Strategy MAX+AVG is c_k-competitive, where c_k satisfies* $c_k = \frac{kx_k^k-(k-1)x_k^{k-1}}{x_k^k-1}$, *with $x_k > 1$ being a zero of the function* $f(x) = (k-1)x^k - k^2x + k^2 - 2k + 1$.

k	c_k
2	1.86603...
3	2.27883...
4	2.49221...
5	2.62163...
6	2.70837...
7	2.77053...
8	2.81724...
9	2.85363...
10	2.88277...
20	3.01425...
40	3.08016...
100	3.11977...
1,000	3.14355...
10,000	3.14592...
100,000	3.14612...
1,000,000	3.14619...

▶ **Theorem 2.** *Strategy MAX+AVG is c_k-competitive, for the values shown in Table 1. Moreover, these values are tight.*

Analysis

Analysis

$$c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$$

Analysis

$$c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$$

$$x_k = \left(1 + \frac{z_k}{k}\right)$$

Analysis

$$c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$$

$$x_k = \left(1 + \frac{z_k}{k}\right)$$

Analysis

$$c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$$

$$x_k = \left(1 + \frac{z_k}{k}\right)$$

$$c_k = \frac{k\left(1 + \frac{z_k}{k}\right) - (k-1)}{\left(1 + \frac{z_k}{k}\right) - \frac{1}{\left(1+\frac{z_k}{k}\right)^{k-1}}}$$

Analysis

$$c_k = \frac{k x_k^k - (k-1) x_k^{k-1}}{x_k^k - 1}$$

$$x_k = \left(1 + \frac{z_k}{k}\right)$$

$$c_k = \frac{k(1+\frac{z_k}{k}) - (k-1)}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}}$$

Analysis

$$c_k = \frac{k(1+\frac{z_k}{k}) - (k-1)}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}}$$

Analysis

$$c_k = \frac{k(1+\frac{z_k}{k}) - (k-1)}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}}$$

$$\lim_{k \to \inf} \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z}{1-e^{-z}}$$

Analysis

$$c_k = \frac{k(1+\frac{z_k}{k}) - (k-1)}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}}$$

$$\lim_{k \to \inf} \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z}{1-e^{-z}}$$

Derivative

Analysis

$$c_k = \frac{k(1+\frac{z_k}{k}) - (k-1)}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}}$$

$$\lim_{k\to\inf} \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z}{1-e^{-z}}$$

Derivative

$$\frac{e^z(-z+e^z-2)}{(e^z-1)^2}$$

Analysis

$$c_k = \frac{k(1+\frac{z_k}{k}) - (k-1)}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}}$$

$$\lim_{k \to \inf} \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z}{1-e^{-z}}$$

Derivative

$$\frac{e^z(-z+e^z-2)}{(e^z-1)^2}$$

Zero of

Analysis

$$c_k = \frac{k(1+\frac{z_k}{k}) - (k-1)}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}}$$

$$\lim_{k\to\inf} \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z}{1-e^{-z}}$$

Derivative

$$\frac{e^z(-z+e^z-2)}{(e^z-1)^2}$$

Zero of

$$e^z = z+2$$

Analysis

$$c_k = \frac{k(1+\frac{z_k}{k}) - (k-1)}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}}$$

$$\lim_{k\to\inf} \frac{1+z_k}{(1+\frac{z_k}{k}) - \frac{1}{(1+\frac{z_k}{k})^{k-1}}} = \frac{1+z}{1-e^{-z}}$$

Derivative

$$\frac{e^z(-z+e^z-2)}{(e^z-1)^2}$$

Zero of

$$e^z = z+2$$

$$c = W_{-1}(-\tfrac{1}{e^2}) = 3.146193220582\ldots$$

Analysis

$$\frac{e^z(-z+e^z-2)}{(e^z-1)^2}$$

$$e^z = z + 2$$

$$c = W_{-1}(-\tfrac{1}{e^2}) = 3.146193220582\ldots$$

Analysis

▸ **Theorem 3.** *Algorithm MAX+AVG is c-competitive for all k, where c is the solution of the equation $e^c = c + 2$. This is the value $W_{-1}(-\frac{1}{e^2}) = 3.146193220582\ldots$, where W_{-1} is the lower branch of Lambert's W-function. Moreover, this is tight: For any $c' < c$, MAX+AVG is not c'-competitive for large enough k.*

$$\frac{e^z(-z+e^z-2)}{(e^z-1)^2}$$

$$e^z = z + 2$$

$$c = W_{-1}(-\tfrac{1}{e^2}) = 3.146193220582\ldots$$

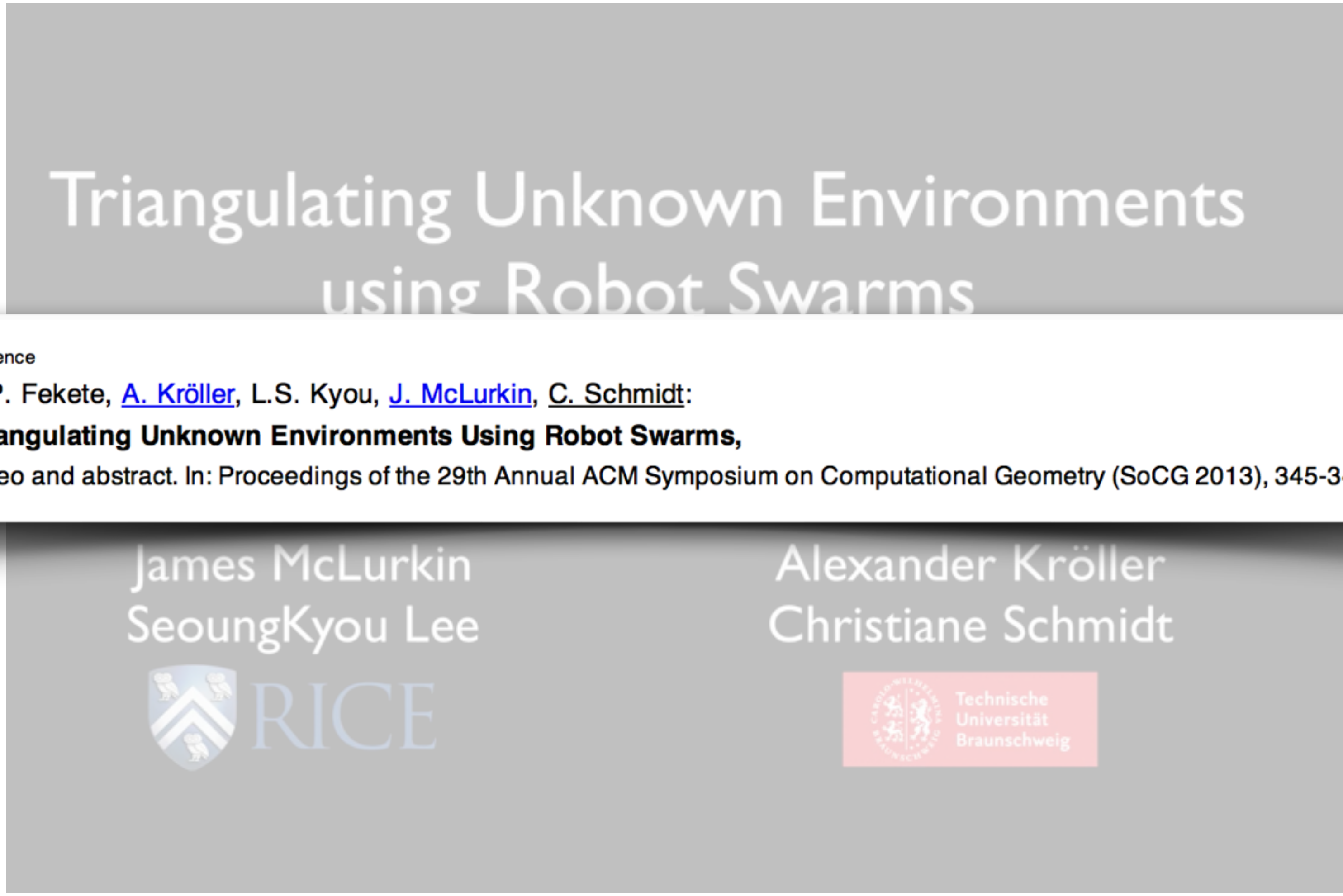
Part 3: Robot Swarms

Part 3.1:
Online Triangulation

Video!


Triangulating Unknown Environments using Robot Swarms
Aaron Becker
James McLurkin
SeoungKyou Lee
RICE
Sándor P. Fekete
Alexander Kröller
Christiane Schmidt
Technische Universität Braunschweig

Video!



conference

S.P. Fekete, A. Kröller, L.S. Kyou, J. McLurkin, C. Schmidt:

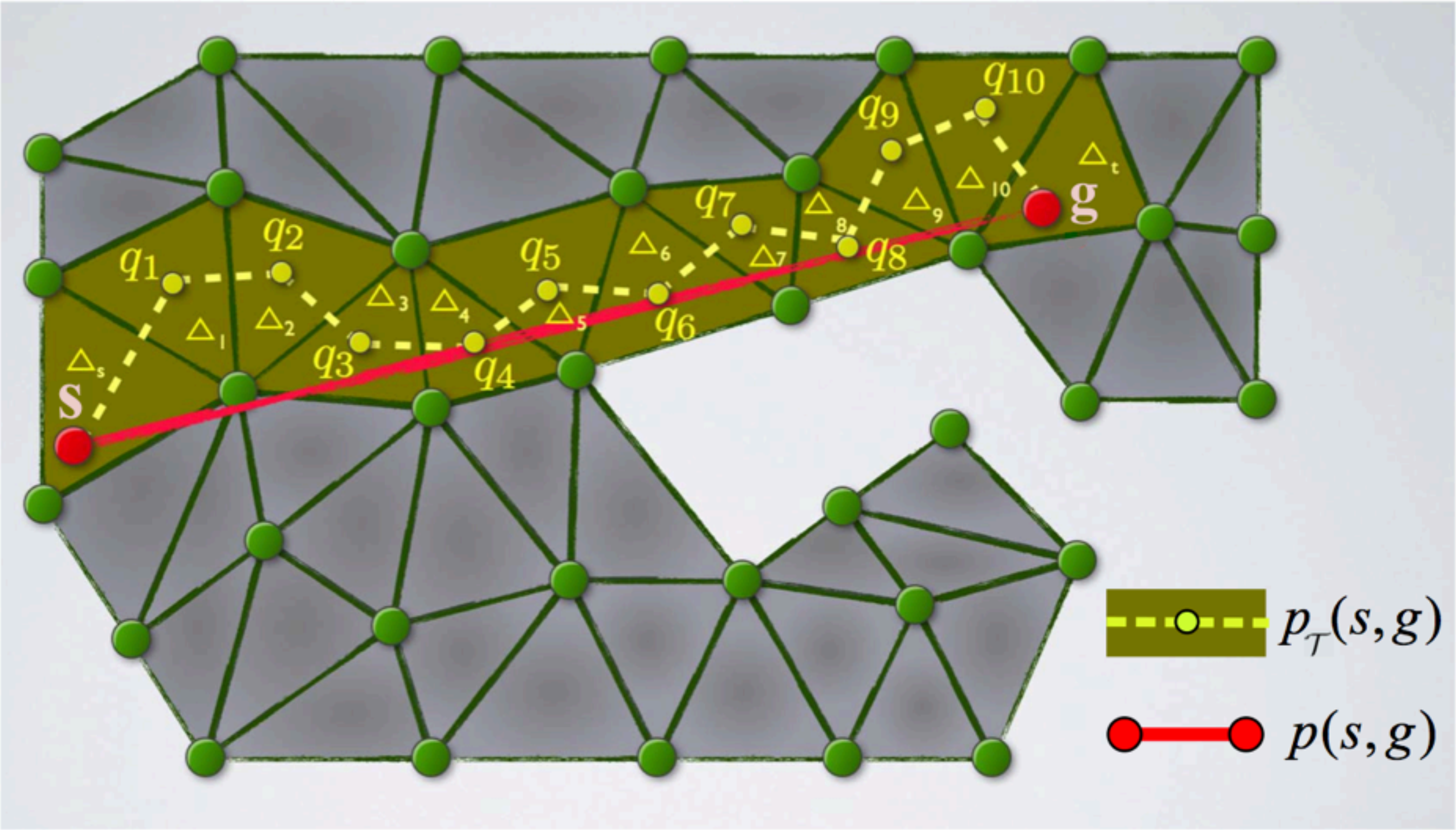
Triangulating Unknown Environments Using Robot Swarms,

Video and abstract. In: Proceedings of the 29th Annual ACM Symposium on Computational Geometry (SoCG 2013), 345-346.

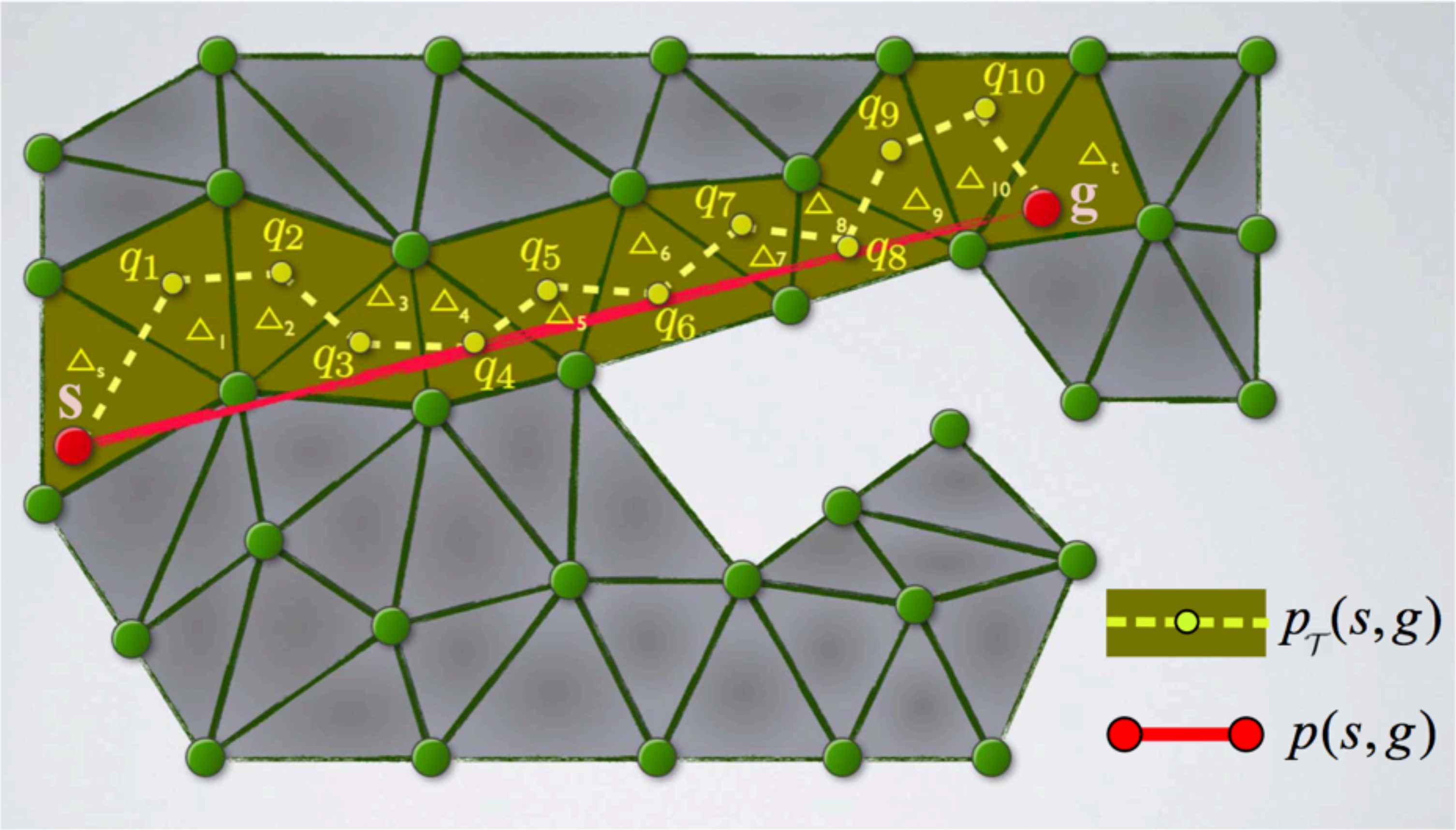
Part 3.2: Local Routing

Dual Routing

Dual Routing

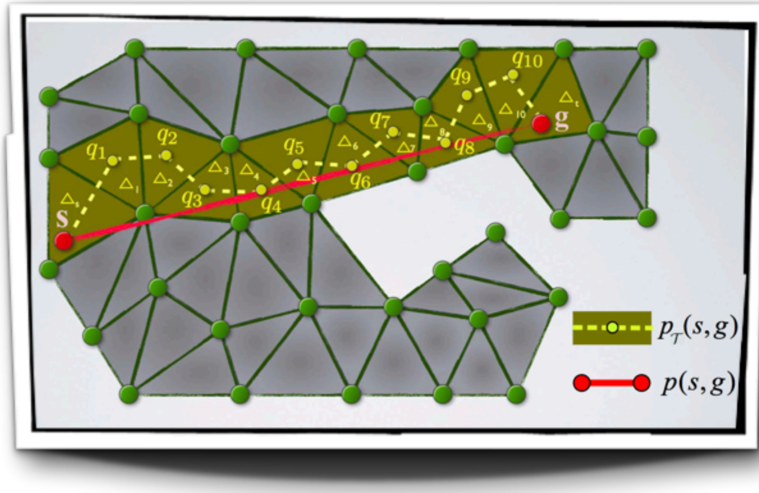


Dual Routing

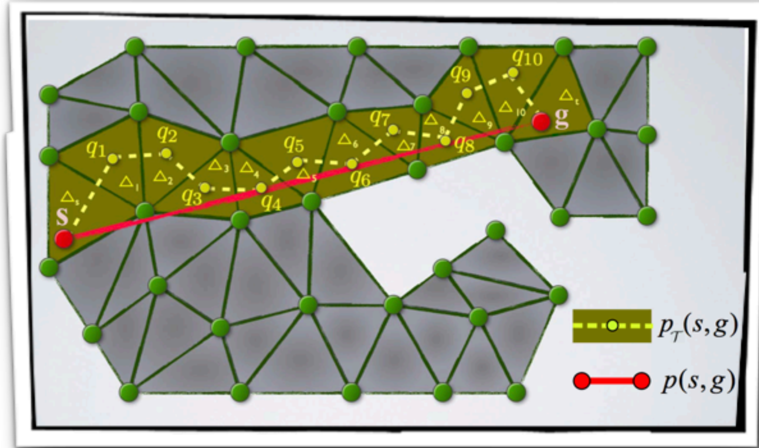


Note*:* The dual graph is stored implicitly in *primal* vertices!

Dual Routing

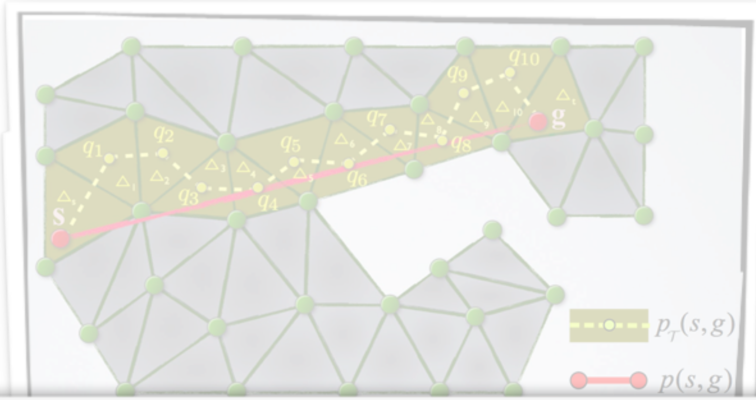


Dual Routing



Theorem 3.3: Consider a (ρ, α)-fat triangulation $\mathcal{T}$ of a planar region $\mathcal{R}$, with vertex set V, maximum and minimum edge length r_{max} and r_{min}, respectively. Let s, g be points in $\mathcal{R}$ that are separated by at least one triangle, i.e., the triangles Δ_s, Δ_g in $\mathcal{T}$ that contain s and g do not share a vertex. Let $p(s,g)$ be a shortest polygonal path in $\mathcal{R}$ that connects s with g, and let $d_p(s,g)$ be its length. Let $p_{\mathcal{T}}(s,g)$ be a $\mathcal{T}$-greedy path between s and g, of length $d_{p_{\mathcal{T}}}(s,g)$. Then $d_{p_{\mathcal{T}}}(s,g) \leq c \cdot d_p(s,g) + 2$, for $c = \lfloor \frac{2\pi}{\alpha} \rfloor \frac{\rho}{\sin(\alpha/2)}$, and $d_{p_{\mathcal{T}}}(s,g) \leq c' \cdot d_p(s,g)$, for $c' = \lfloor \frac{6\pi}{\alpha} \rfloor \frac{\rho}{\sin(\alpha/2)}$.

Dual Routing



conference

S. K. Lee, A. Becker, S.P. Fekete, A. Kröller, J. McLurkin:

Exploration via Structured Triangulation by a Multi-Robot System with Bearing-Only Low-Resolution Sensors,

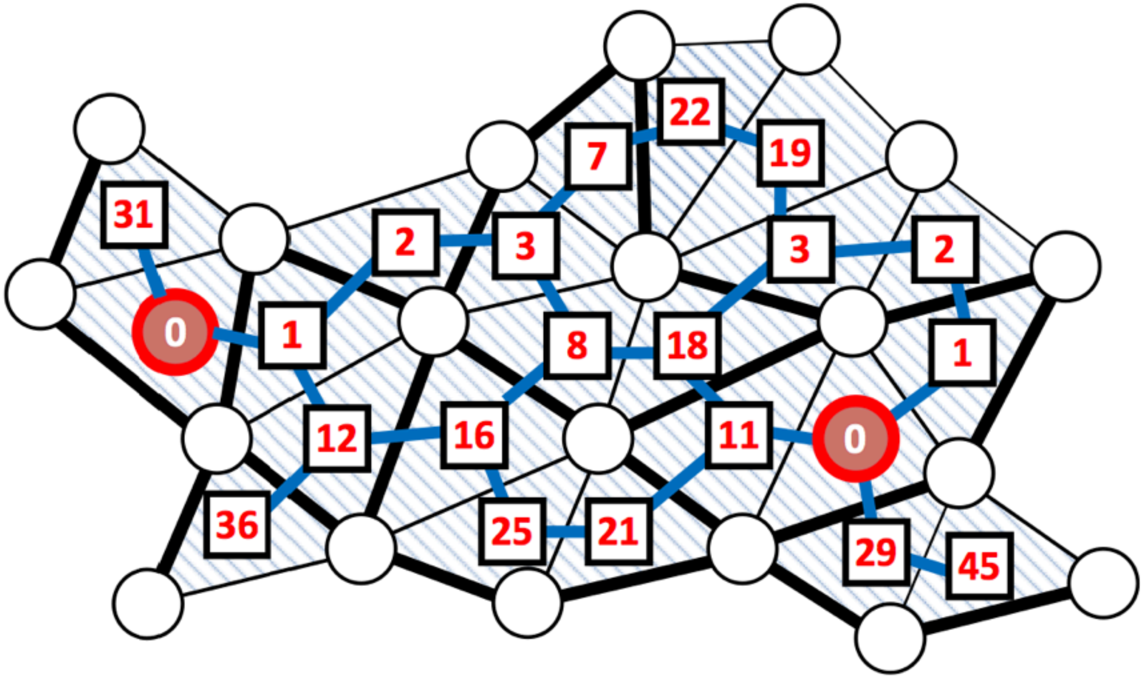
NEW To appear in: 2014 IEEE International Conference on Robotics and Automation (ICRA 2014)

in $\mathcal{R}$ that are separated by at least one triangle, i.e., the triangles Δ_s, Δ_g in $\mathcal{T}$ that contain s and g do not share a vertex. Let $p(s,g)$ be a shortest polygonal path in $\mathcal{R}$ that connects s with g, and let $d_p(s,g)$ be its length. Let $p_{\mathcal{T}}(s,g)$ be a $\mathcal{T}$-greedy path between s and g, of length $d_{p_{\mathcal{T}}}(s,g)$. Then $d_{p_{\mathcal{T}}}(s,g) \leq c \cdot d_p(s,g) + 2$, for $c = \lfloor \frac{2\pi}{\alpha} \rfloor \frac{\rho}{\sin(\alpha/2)}$, and $d_{p_{\mathcal{T}}}(s,g) \leq c' \cdot d_p(s,g)$, for $c' = \lfloor \frac{6\pi}{\alpha} \rfloor \frac{\rho}{\sin(\alpha/2)}$.

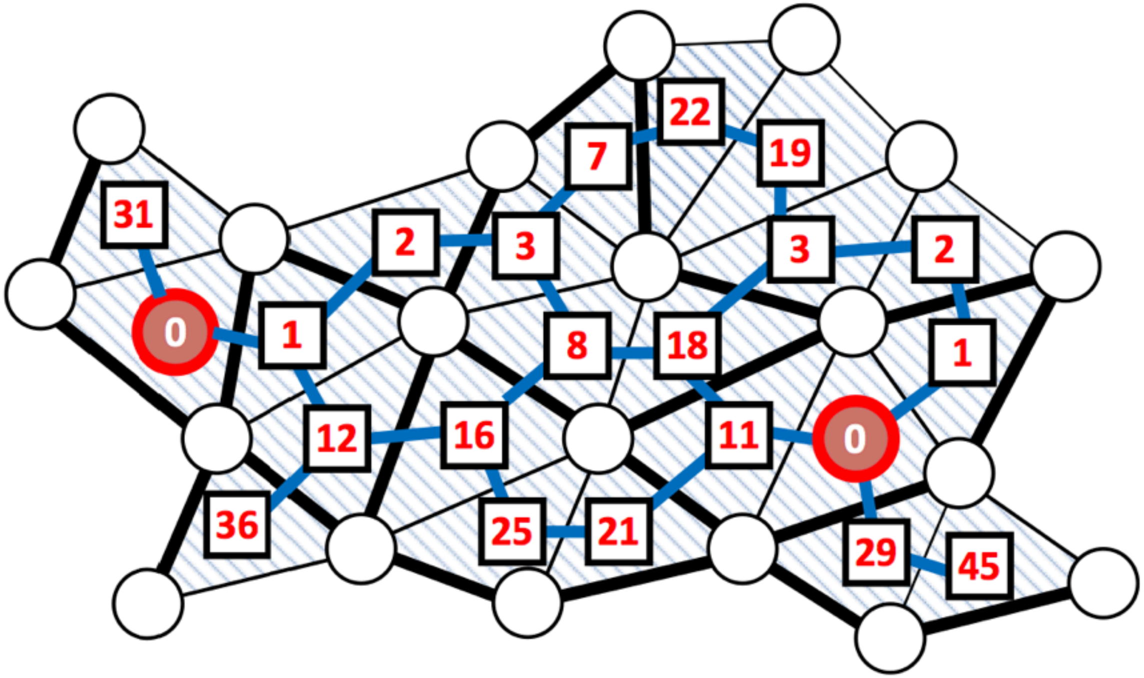
Part 3.3: Local Patrolling Policies

Time Stamps in the Dual Graph

Time Stamps in the Dual Graph

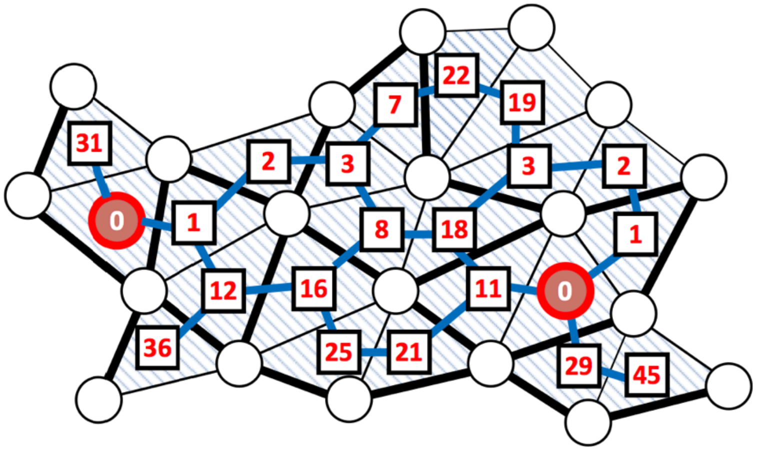


Time Stamps in the Dual Graph

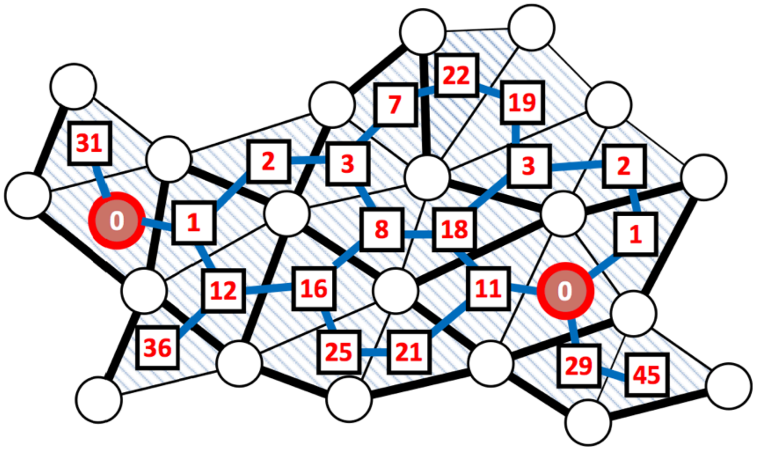


Numbers: Time of last visit

Least Recently Visited

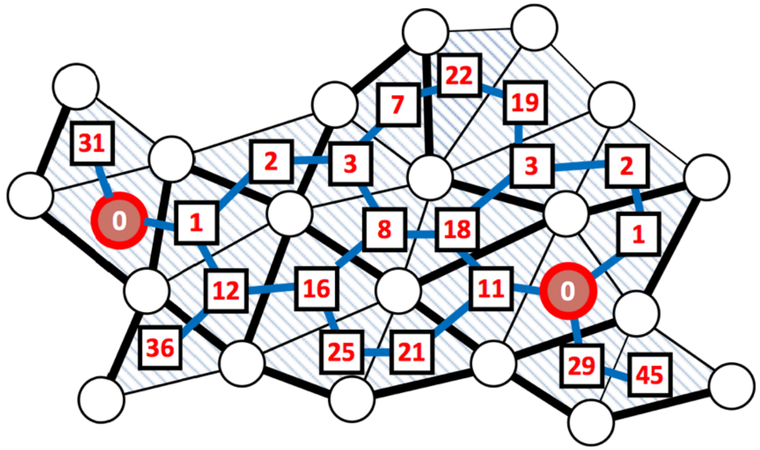


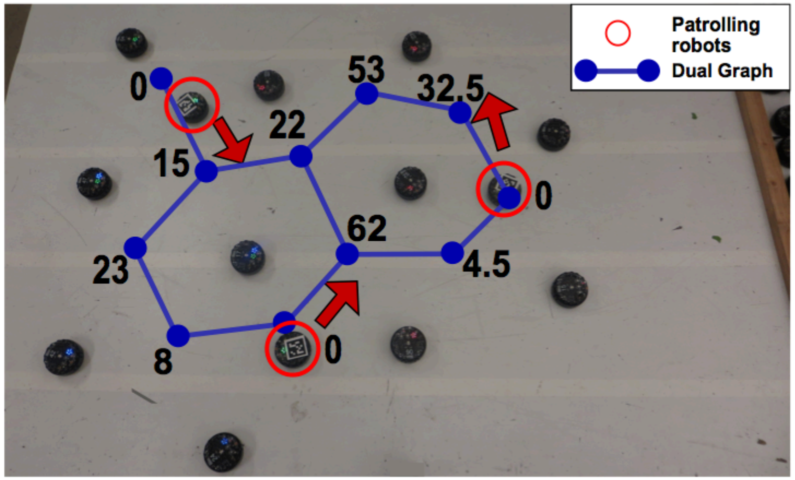
Least Recently Visited



Least Recently Visited (LRV):
Move to vertex with oldest time stamp

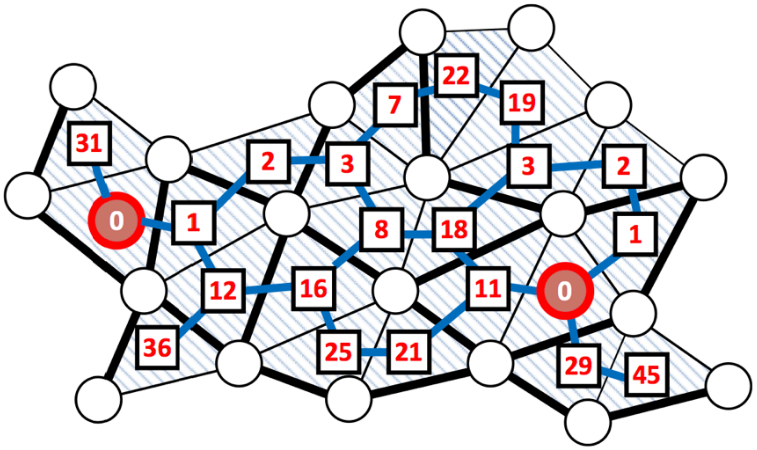
Least Recently Visited

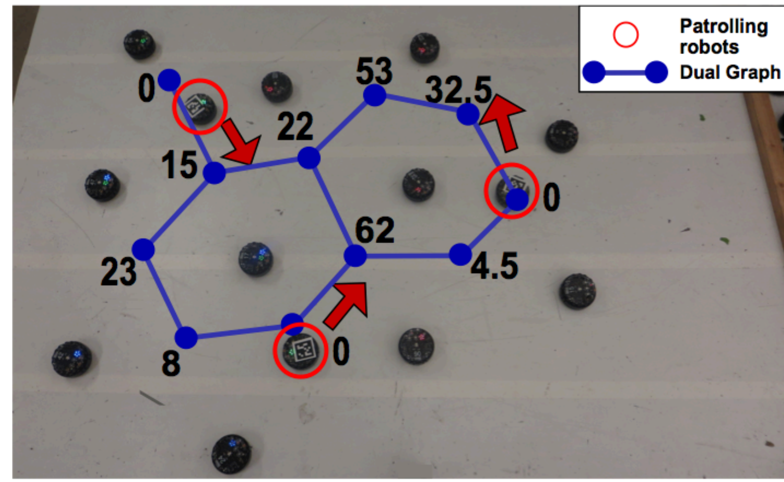




Least Recently Visited (LRV):
Move to vertex with oldest time stamp

Least Recently Visited

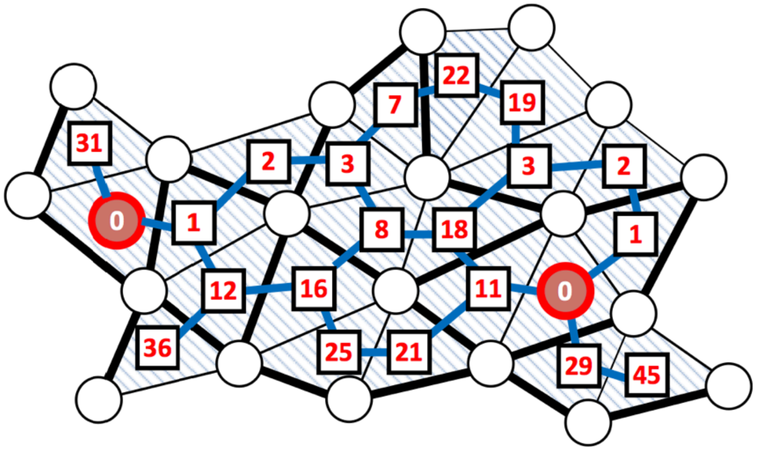


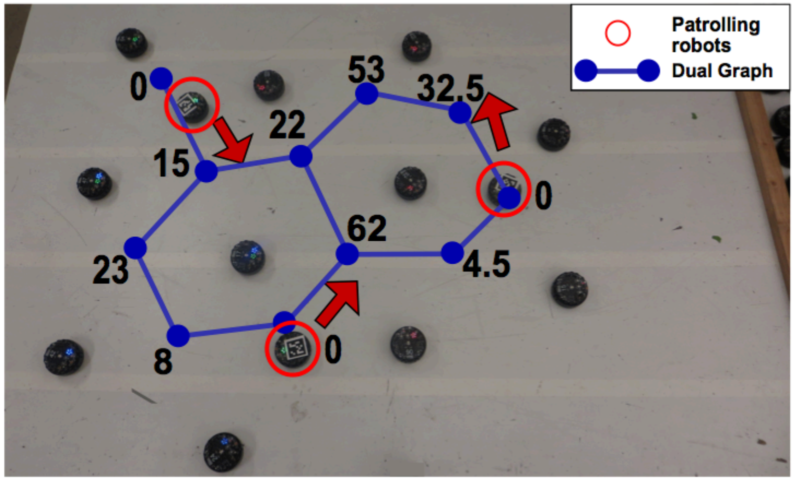


Least Recently Visited (LRV):
Move to vertex with oldest time stamp

Good news: LRV achieves full coverage.

Least Recently Visited



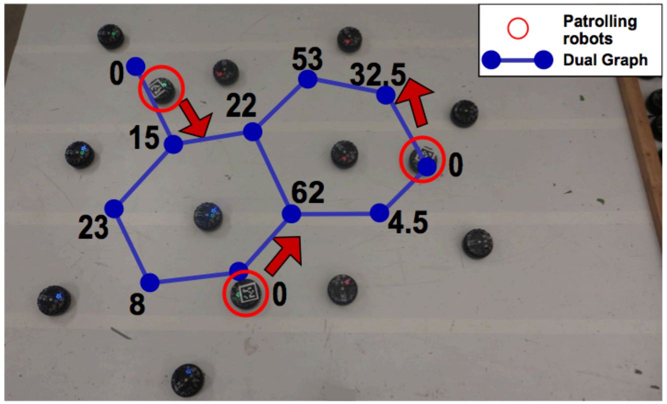


Least Recently Visited (LRV):
Move to vertex with oldest time stamp

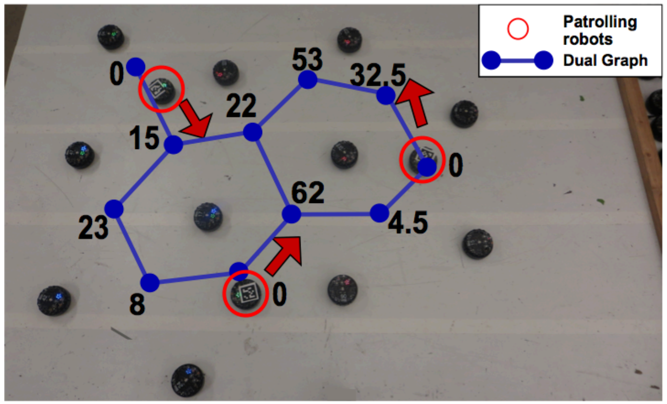
Good news: LRV achieves full coverage.

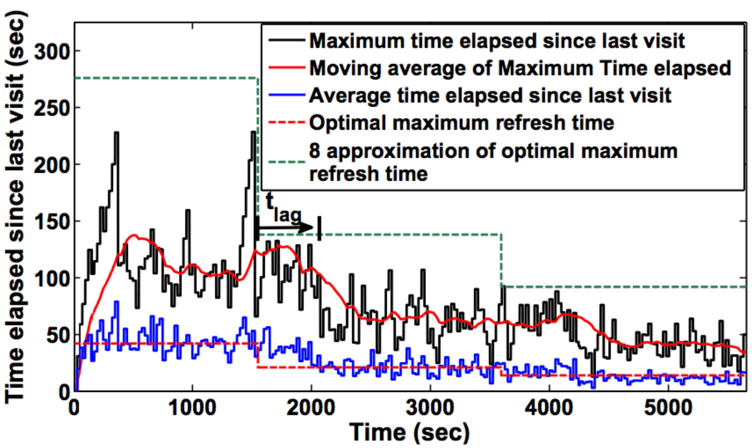
Bad news: The coverage time of LRV can be exponentially large.

LRV: Experimental Results

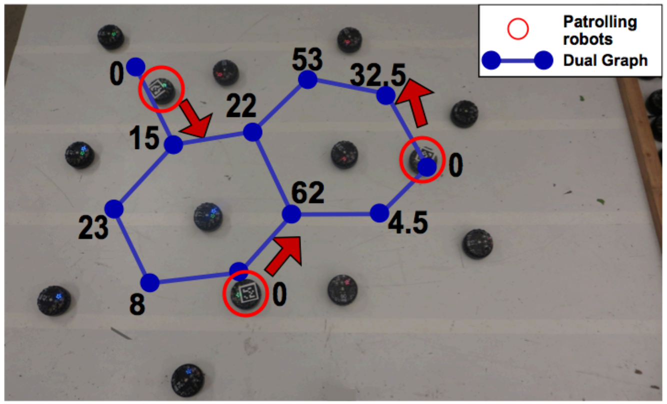


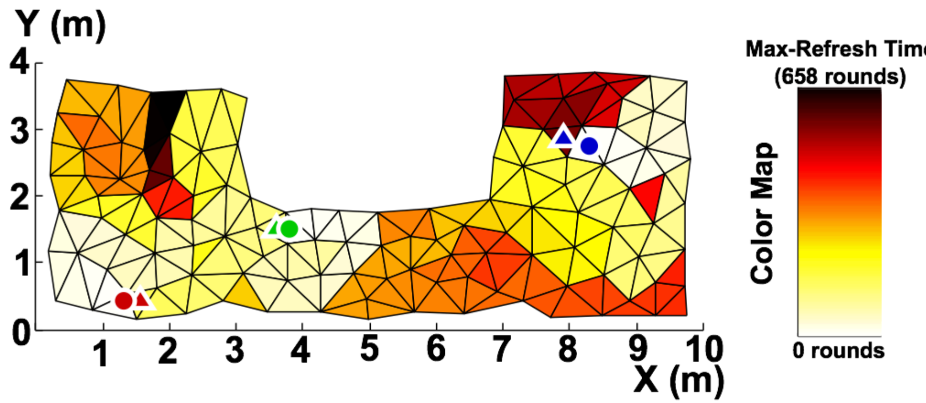
LRV: Experimental Results

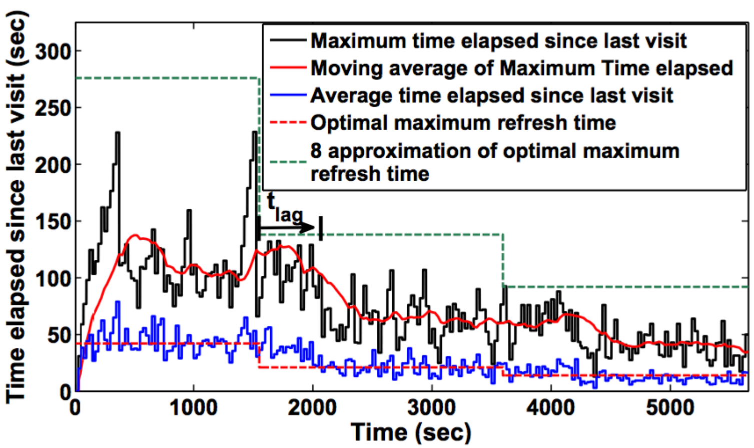




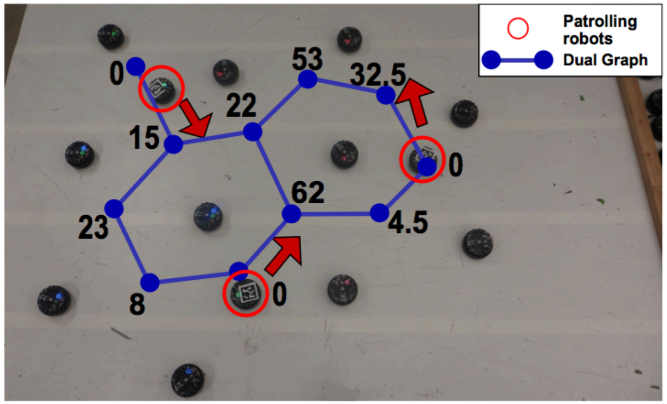
LRV: Experimental Results

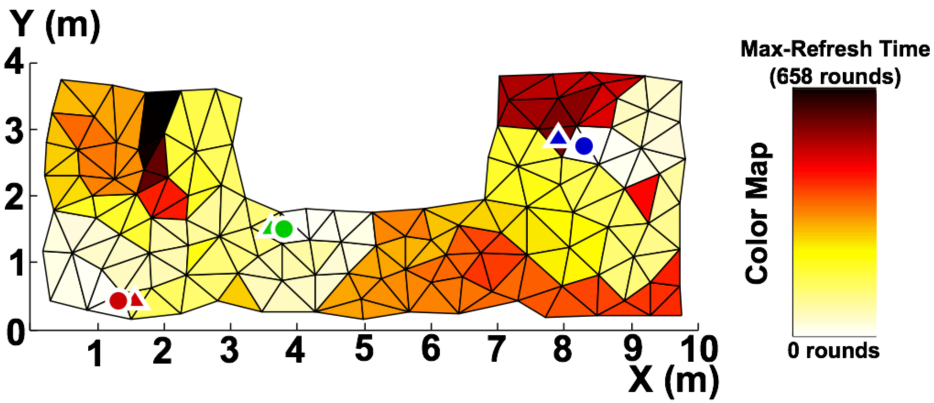


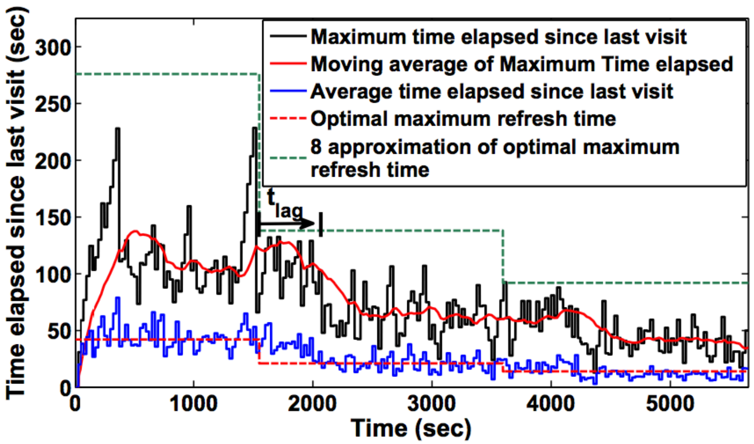


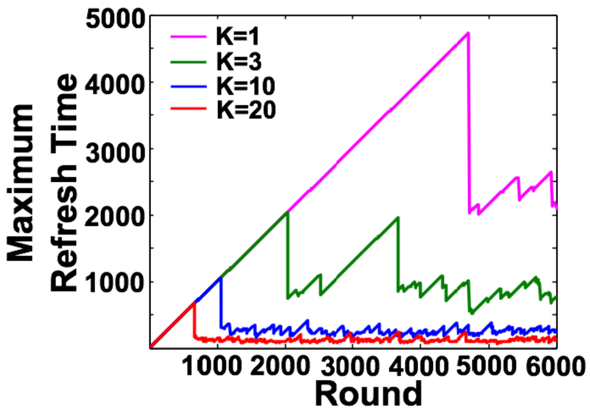


LRV: Experimental Results









Part 4:
Controlling Massive Particle Swarms

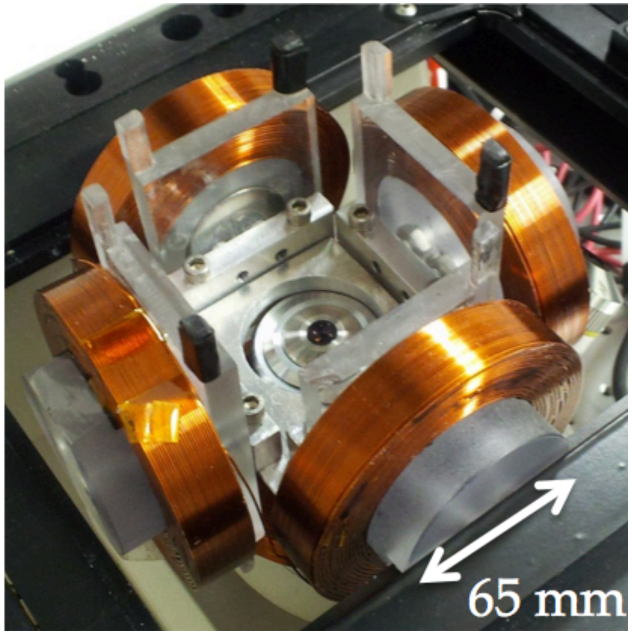
Moving Small Objects

Moving Small Objects

Moving Small Objects

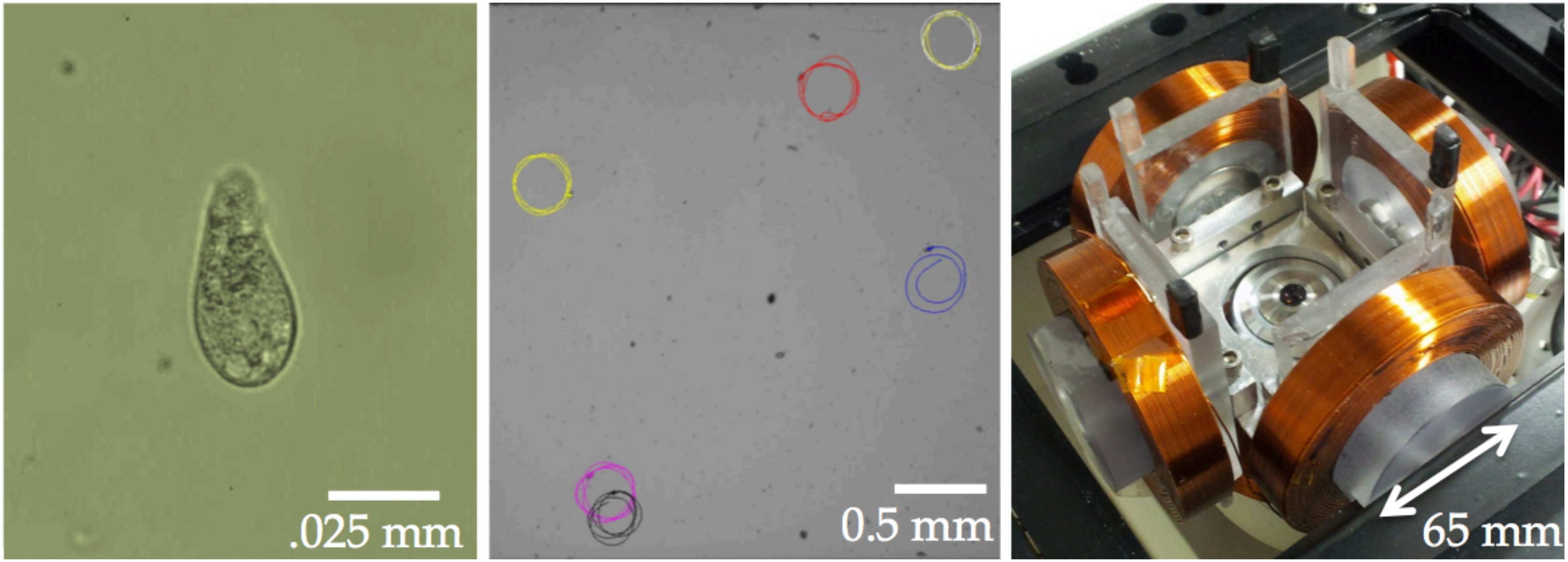
Tetrahymena pyriformis

Moving Small Objects



Tetrahymena pyriformis

Moving Small Objects



Tetrahymena pyriformis

This Part

This Part

- Massive particle swarms
- Global control, not individual motion

This Part

- Massive particle swarms
- Global control, not individual motion
    - *We show hardness for given, external obstacles*

This Part

- Massive particle swarms
- Global control, not individual motion
  - *We show hardness for given, external obstacles*
  - *We establish positive results for designed, additional obstacles*

This Part

- Massive particle swarms
- Global control, not individual motion
  - *We show hardness for given, external obstacles*
  - *We establish positive results for designed, additional obstacles*
- Work in progress, combining theory and practice

This Part

- Massive particle swarms
- Global control, not individual motion
  - *We show hardness for given, external obstacles*
  - *We establish positive results for designed, additional obstacles*
- Work in progress, combining theory and practice

This Part

- Massive particle swarms

conference

A. Becker, E.D. Demaine, S.P. Fekete, G. Habibi, J. McLurkin:
Reconfiguring Massive Particle Swarms with Limited, Global Control,
NEW In: ALGOSENSORS 2013, pp. 51-66, Springer LNCS 8343, 2014.

  - *We establish positive results for designed, additional obstacles*
- Work in progress, combining theory and practice

This Part

- Massive particle swarms

conference

A. Becker, E.D. Demaine, S.P. Fekete, G. Habibi, J. McLurkin:
Reconfiguring Massive Particle Swarms with Limited, Global Control,
NEW In: ALGOSENSORS 2013, pp. 51-66, Springer LNCS 8343, 2014.

- *We establish positive results for*

conference

A. Becker, E.D. Demaine, S.P. Fekete, J. McLurkin:
Particle Computation: Controlling Robot Swarms with only Global Signals,
NEW To appear in: 2014 IEEE International Conference on Robotics and Automation (ICRA 2014)

combining theory and practice

Part 4.1: Why Obstacles Are a Nuisance

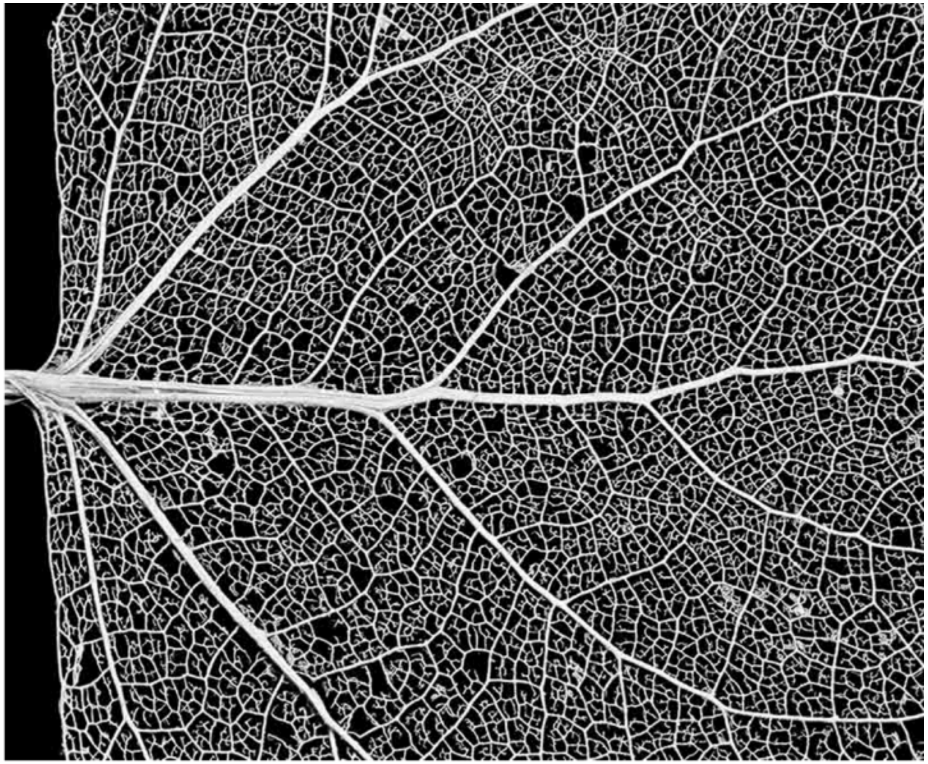
Obstacles as Opponents

Obstacles as Opponents

- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.

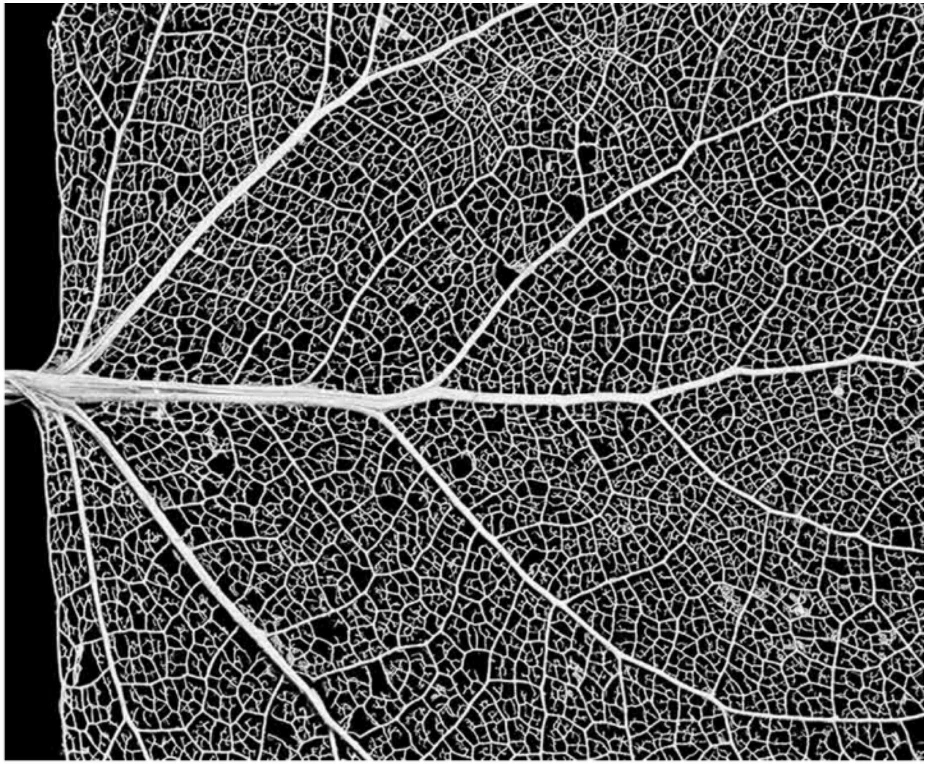
Obstacles as Opponents

- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.



Obstacles as Opponents

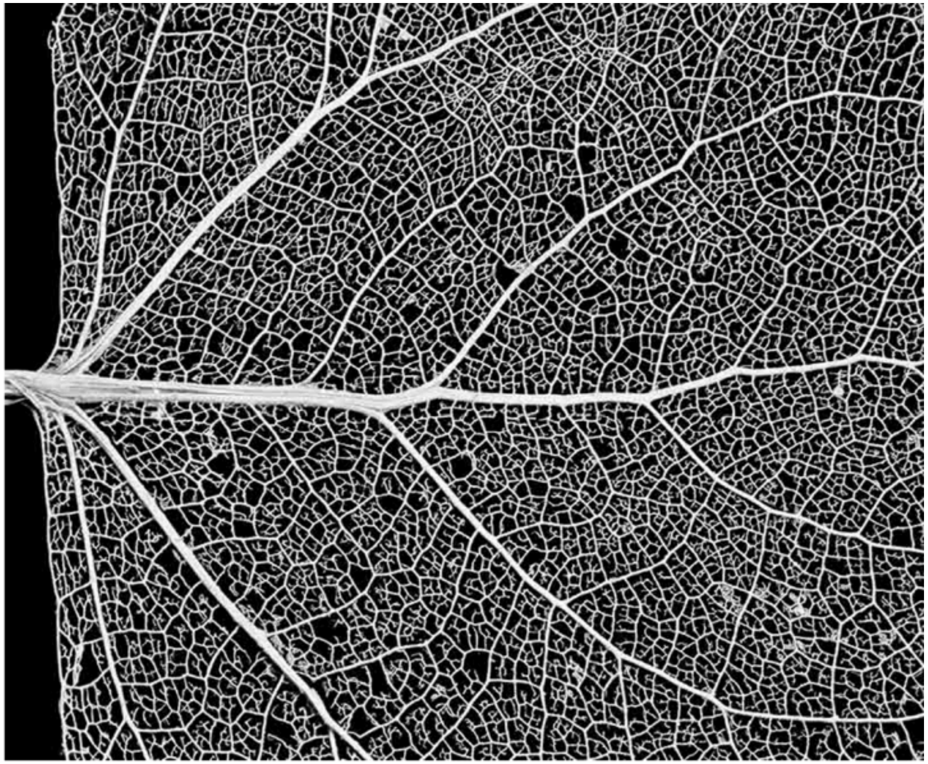
- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.



Cottonwood leaf vascular network

Obstacles as Opponents

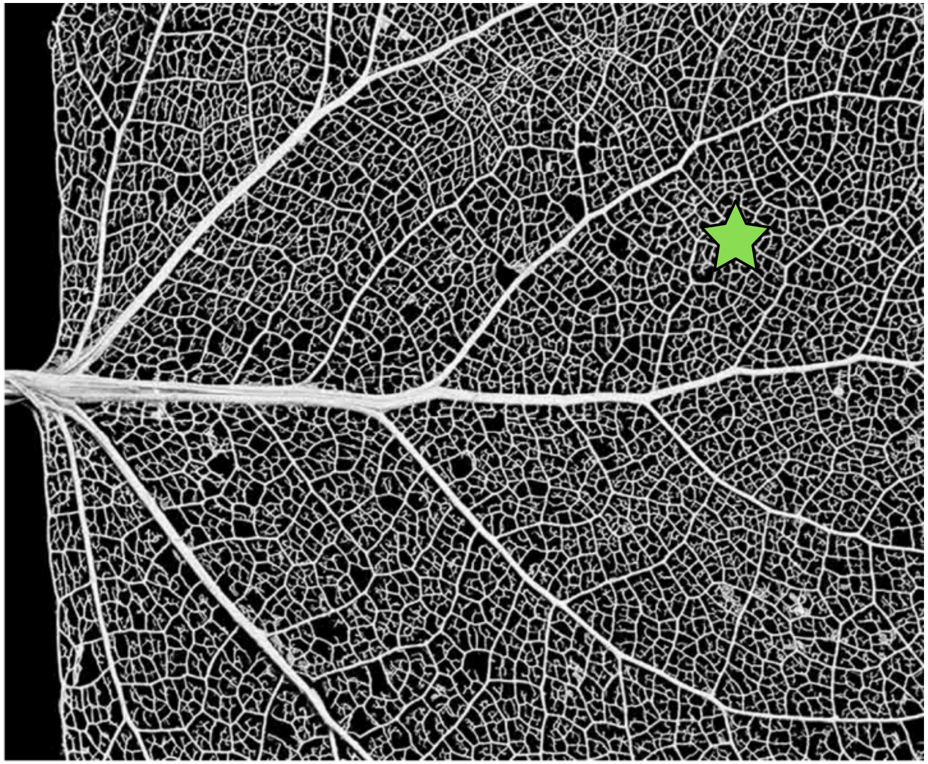
- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.



Cottonwood leaf vascular network

Obstacles as Opponents

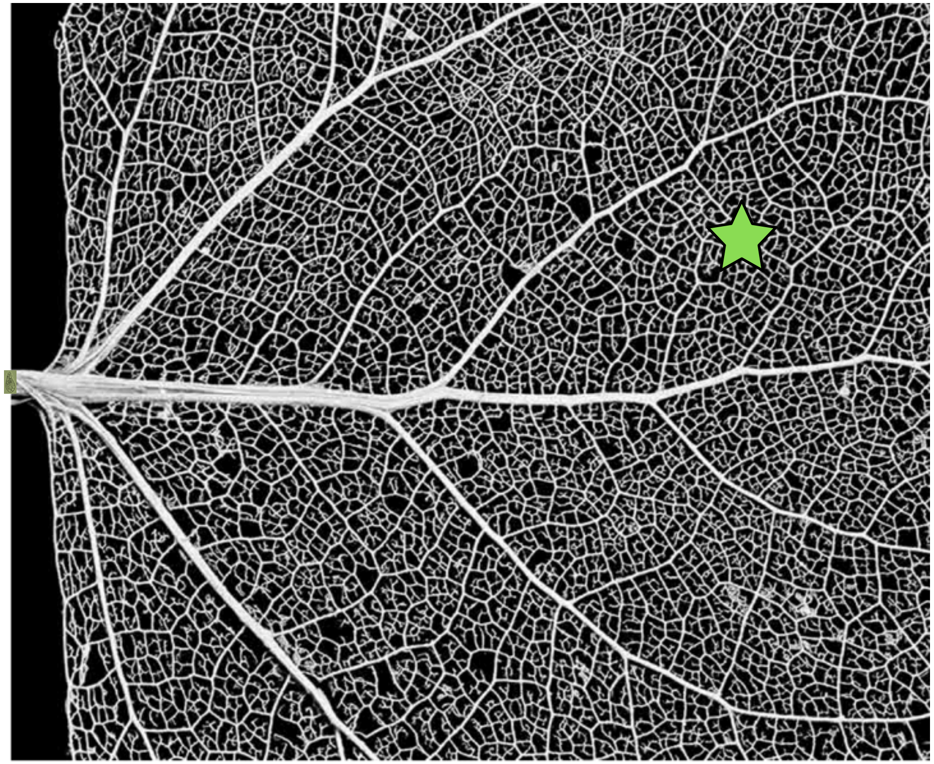
- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.



Cottonwood leaf vascular network

Obstacles as Opponents

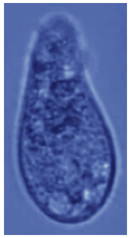
- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.

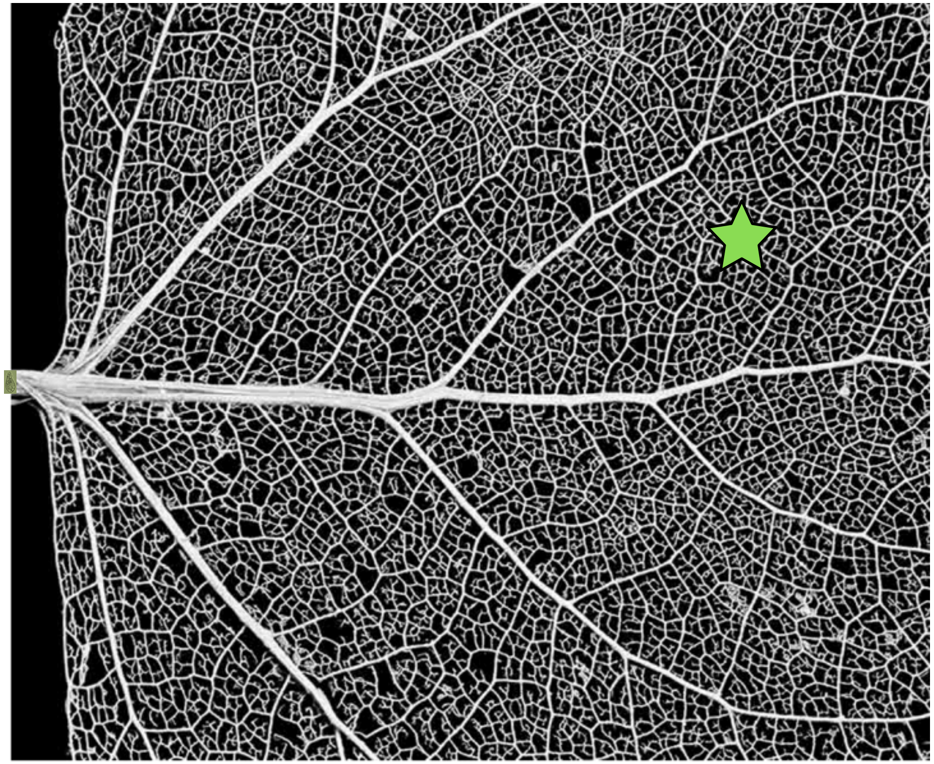


Cottonwood leaf vascular network

Obstacles as Opponents

- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.

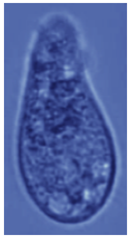


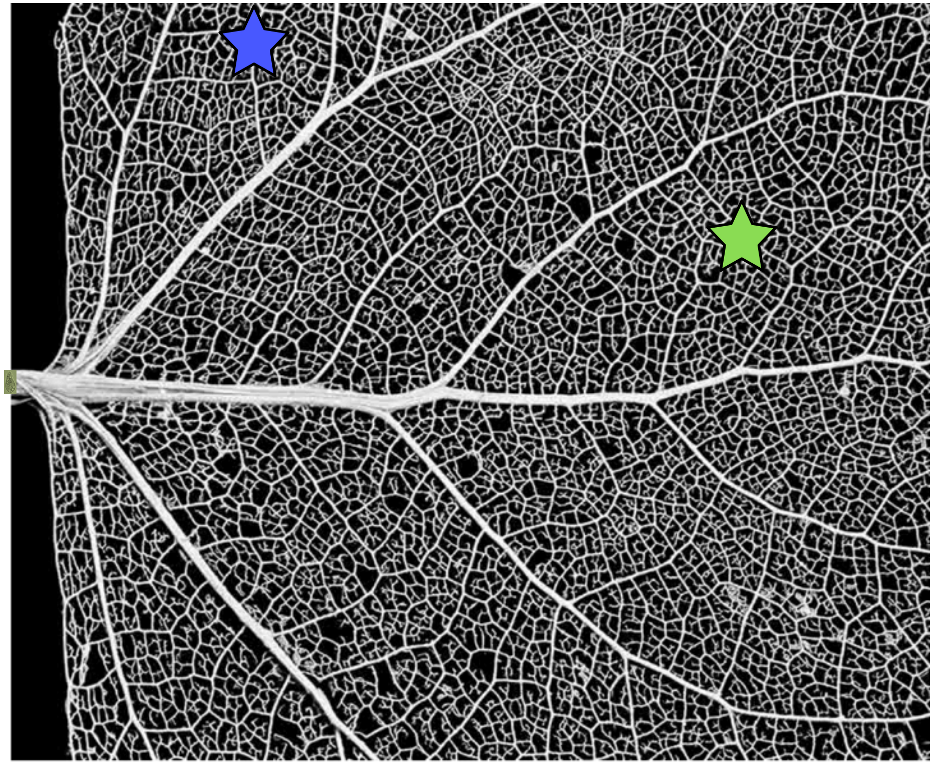


Cottonwood leaf vascular network

Obstacles as Opponents

- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.

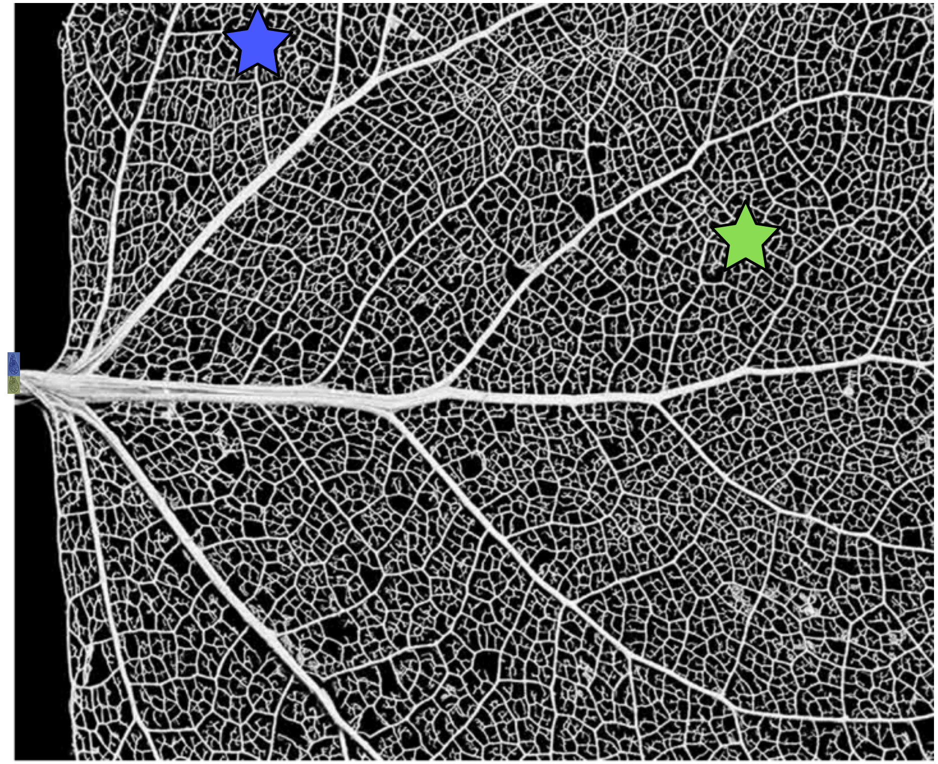




Cottonwood leaf vascular network

Obstacles as Opponents

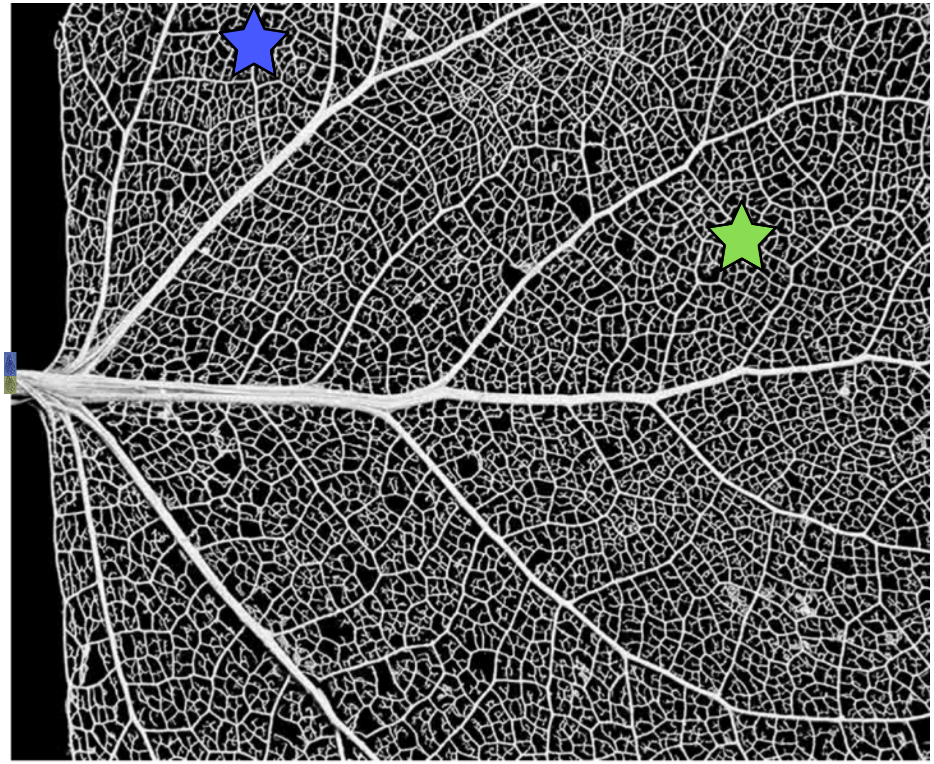
- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.



Cottonwood leaf vascular network

Obstacles as Opponents

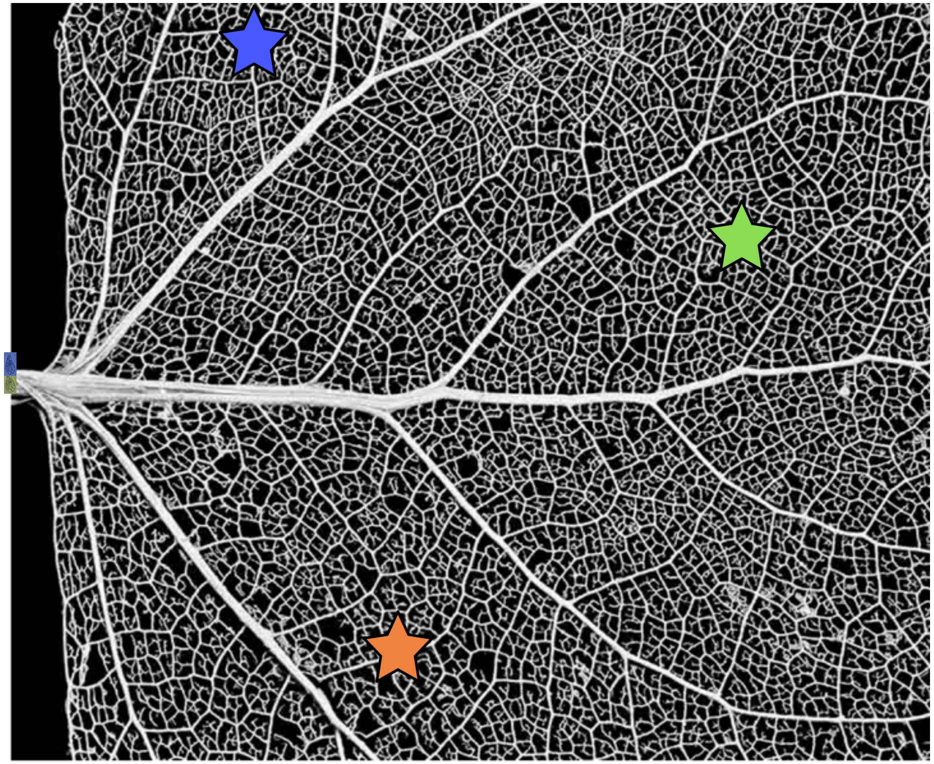
- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.



Cottonwood leaf vascular network

Obstacles as Opponents

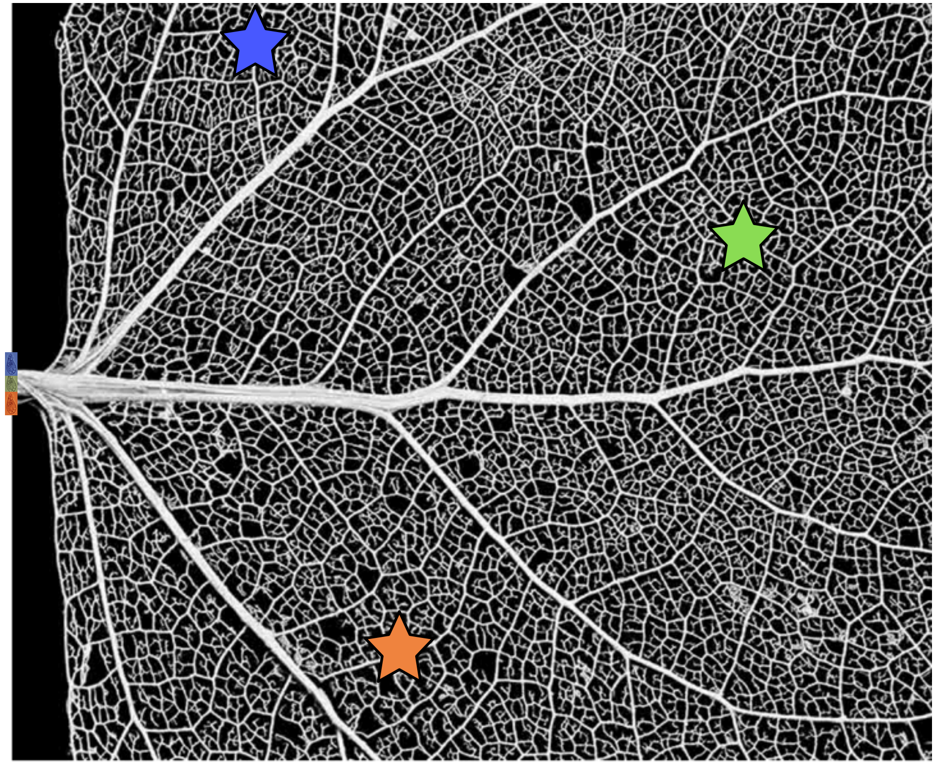
- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.



Cottonwood leaf vascular network

Obstacles as Opponents

- Targets may not be easy to reach.
- Motion planning gets quite tricky in parallel.

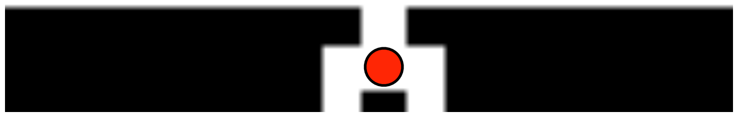


Cottonwood leaf vascular network

Complexity: Binary Variables

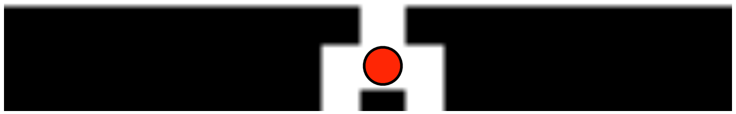
Complexity: Binary Variables

Complexity: Binary Variables



Complexity: Binary Variables

Choice: left or right?
Independent choices?!



Complexity: Binary Variables

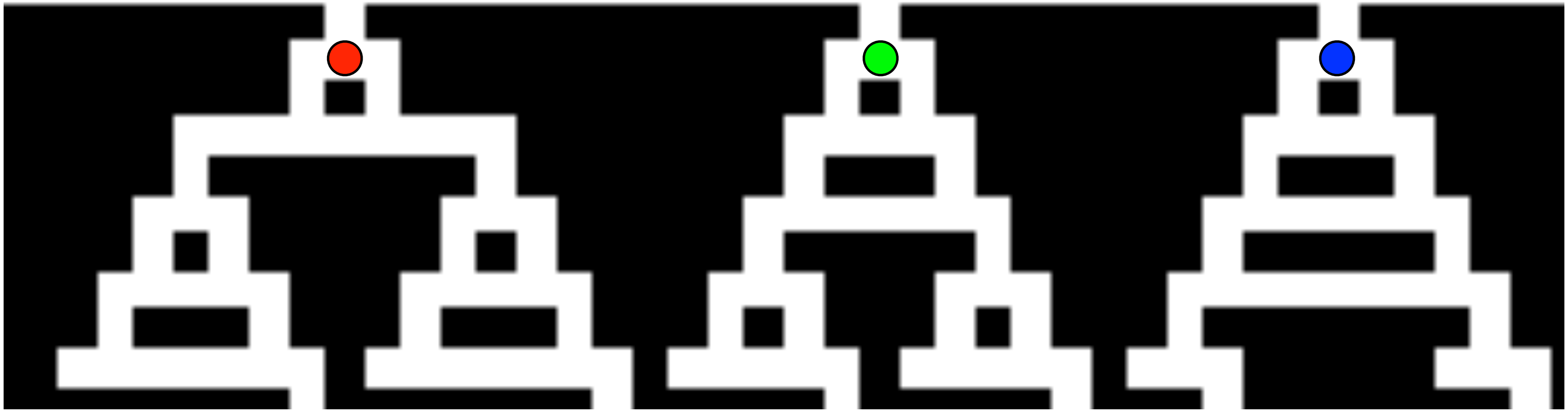
Choice: left or right?
Independent choices?!

Complexity: Binary Variables

Choice: left or right?
Independent choices?!

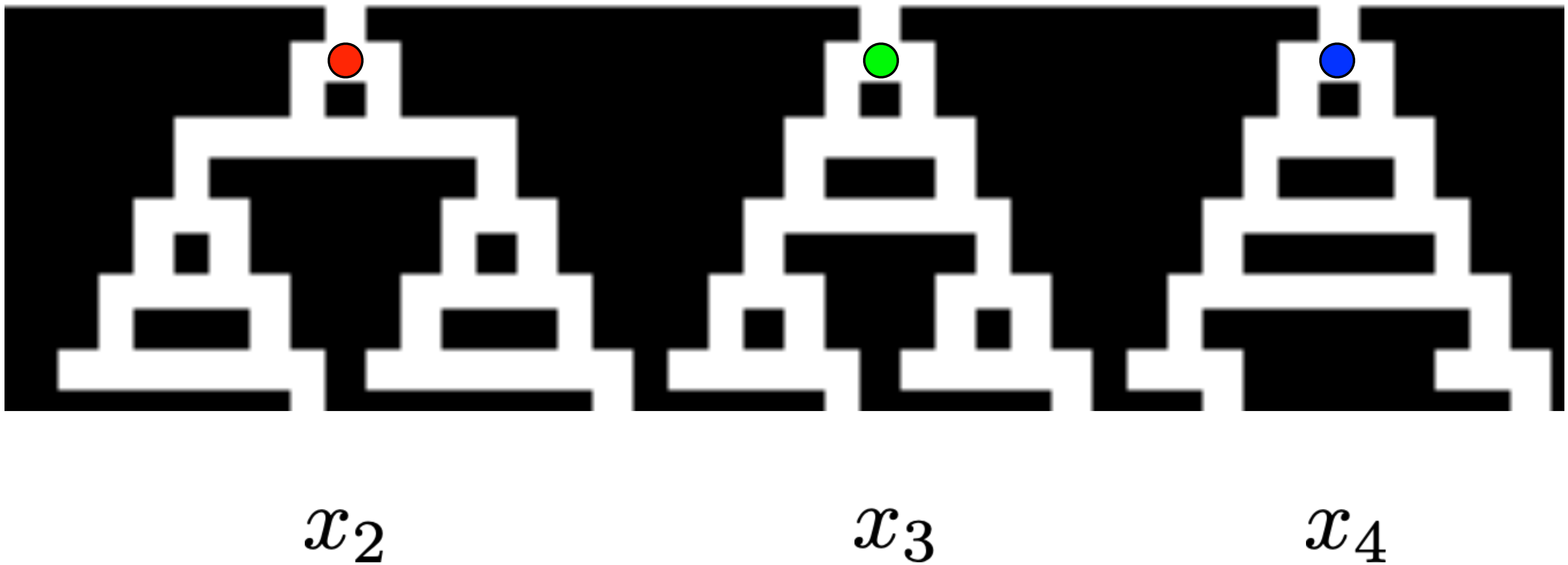
Complexity: Binary Variables

Choice: left or right?
Independent choices?!



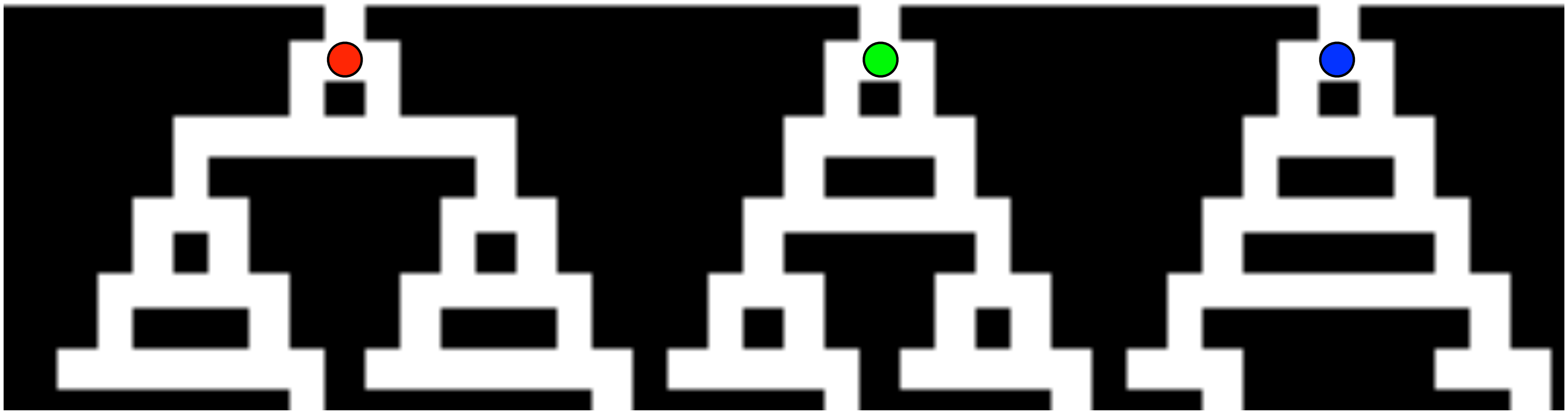
Complexity: Binary Variables

Choice: left or right?
Independent choices?!



Complexity: Binary Variables

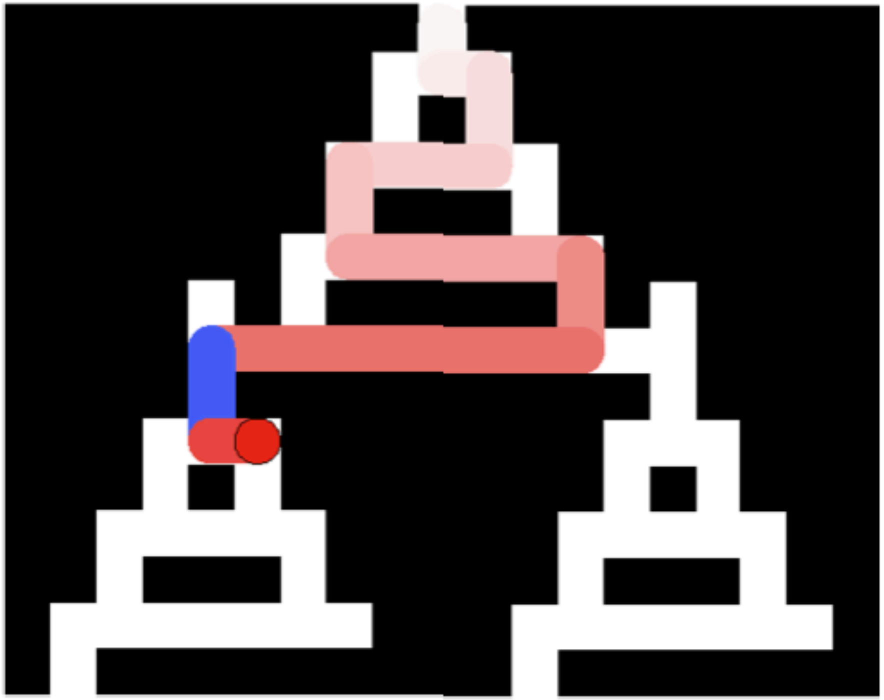
Choice: left or right?
Independent choices?!



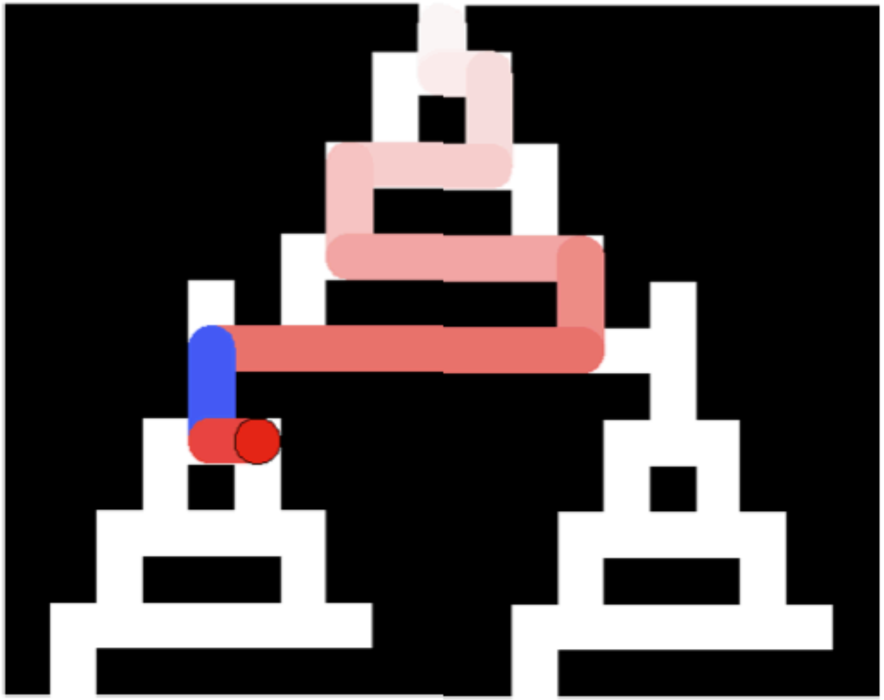
x_2 x_3 x_4

Choice only matters when it is a variable's "turn"!

Complexity: Binary Variables



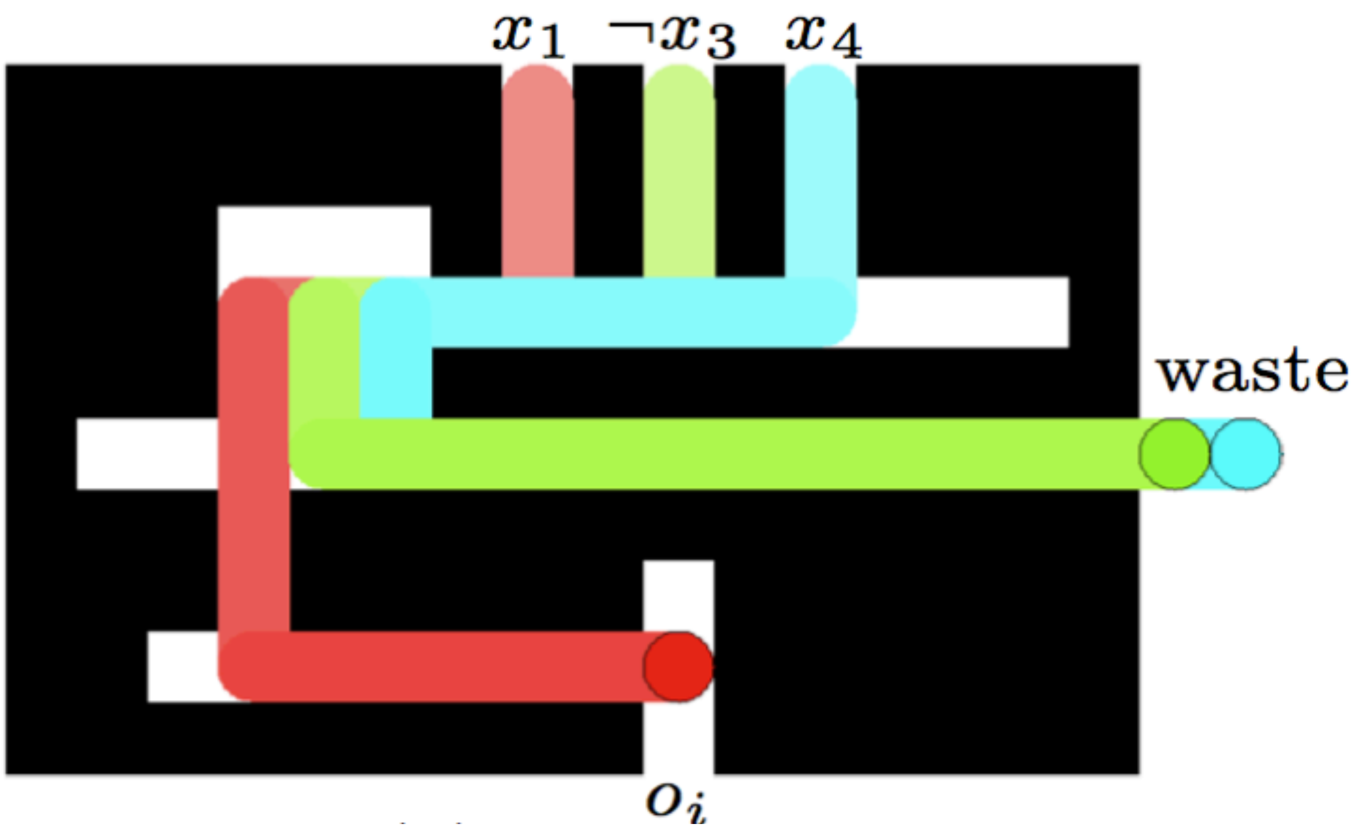
Complexity: Binary Variables



Minor detail: Avoid reversible choices!

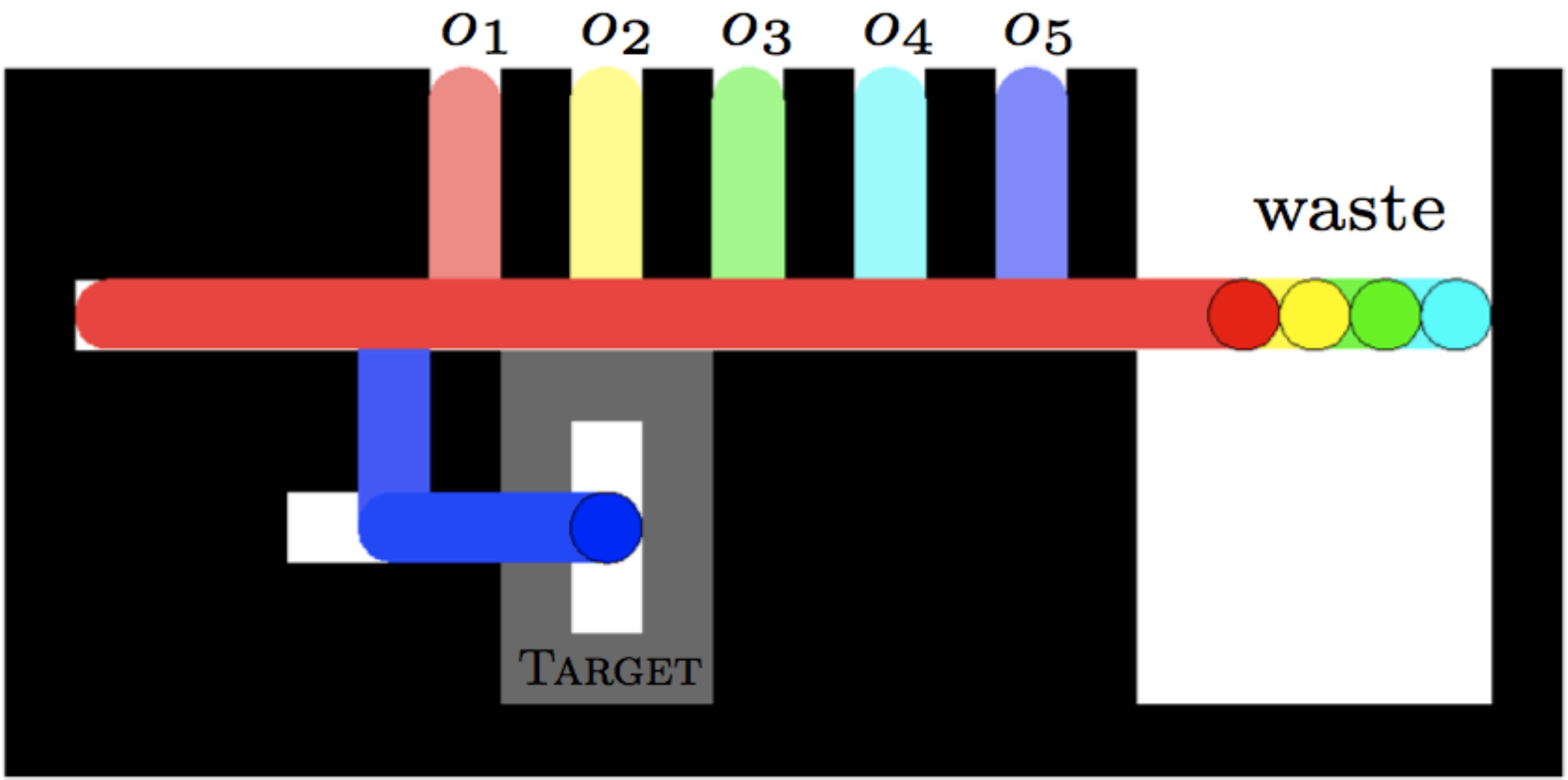
Complexity: Clauses

Complexity: Clauses



Complexity: Truth Checking

Complexity: Truth Checking



Complexity: Overall Construction

$$(\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_3)$$

Complexity: Overall Construction

$(\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_3)$

$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$

Complexity: Overall Construction

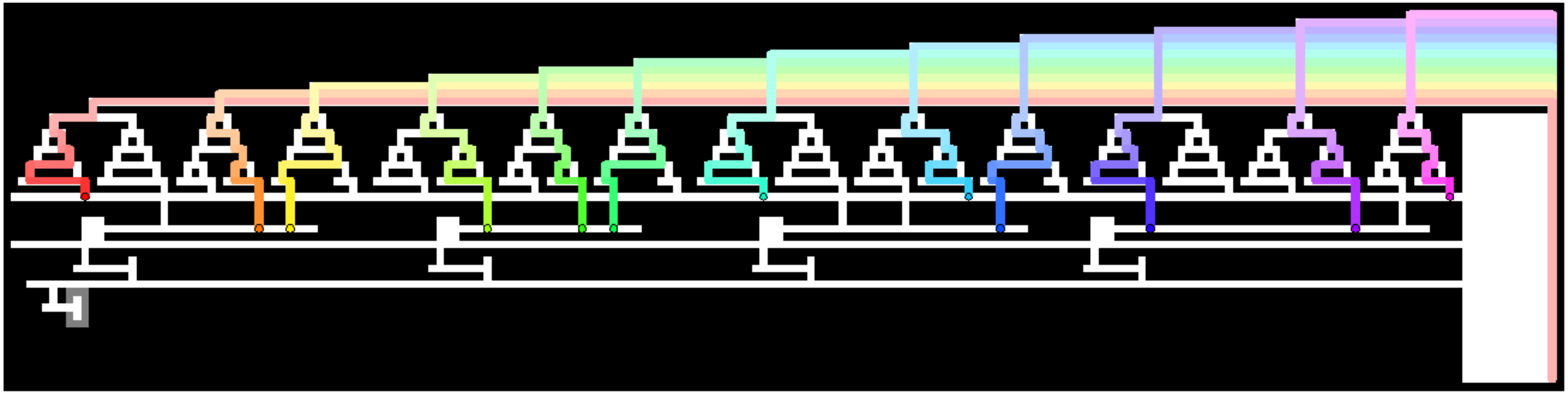
$$(\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_3)$$

$$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$$

Complexity: Overall Construction

$(\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_3)$

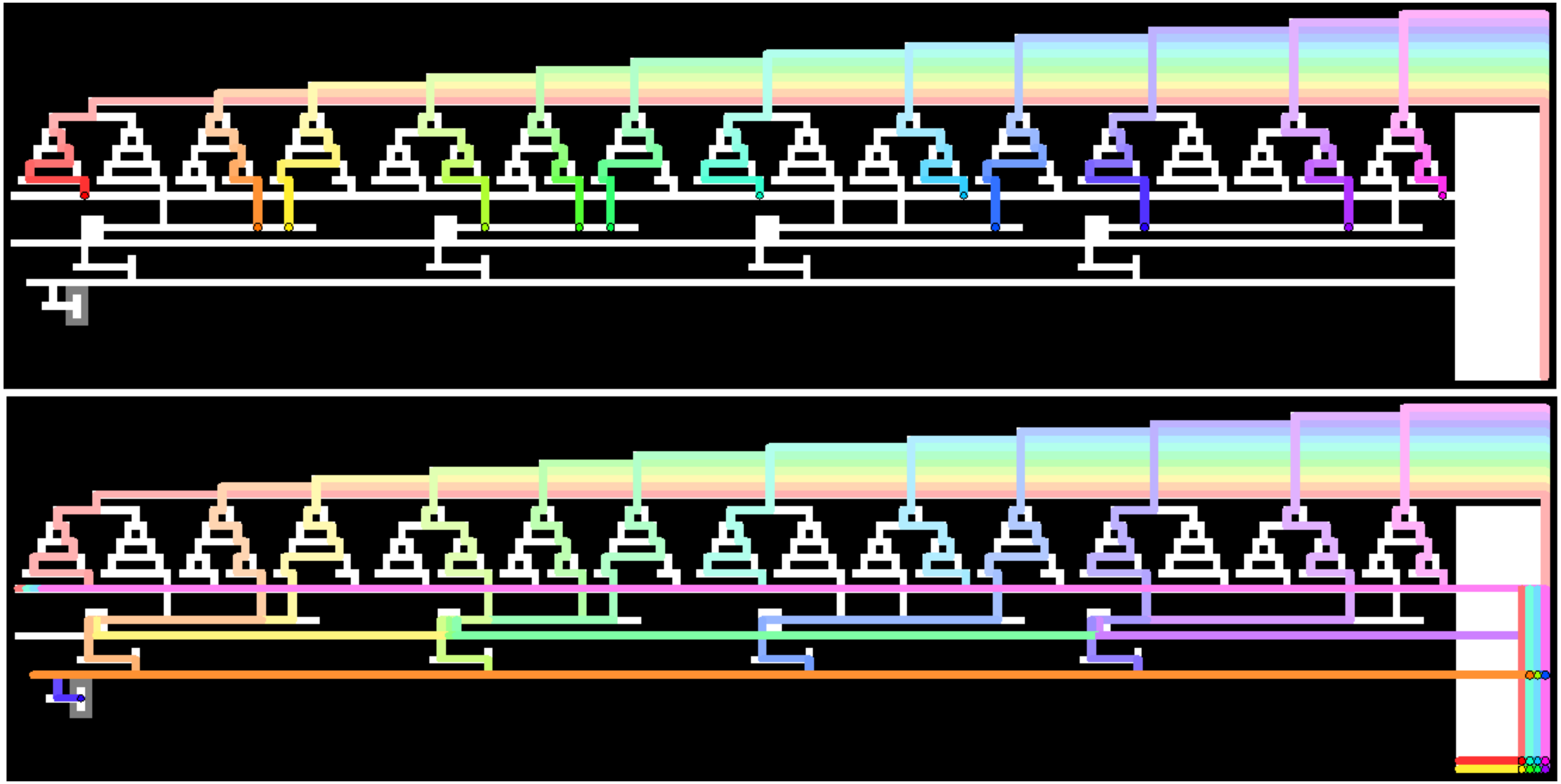
$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$



Complexity: Overall Construction

$(\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_3)$

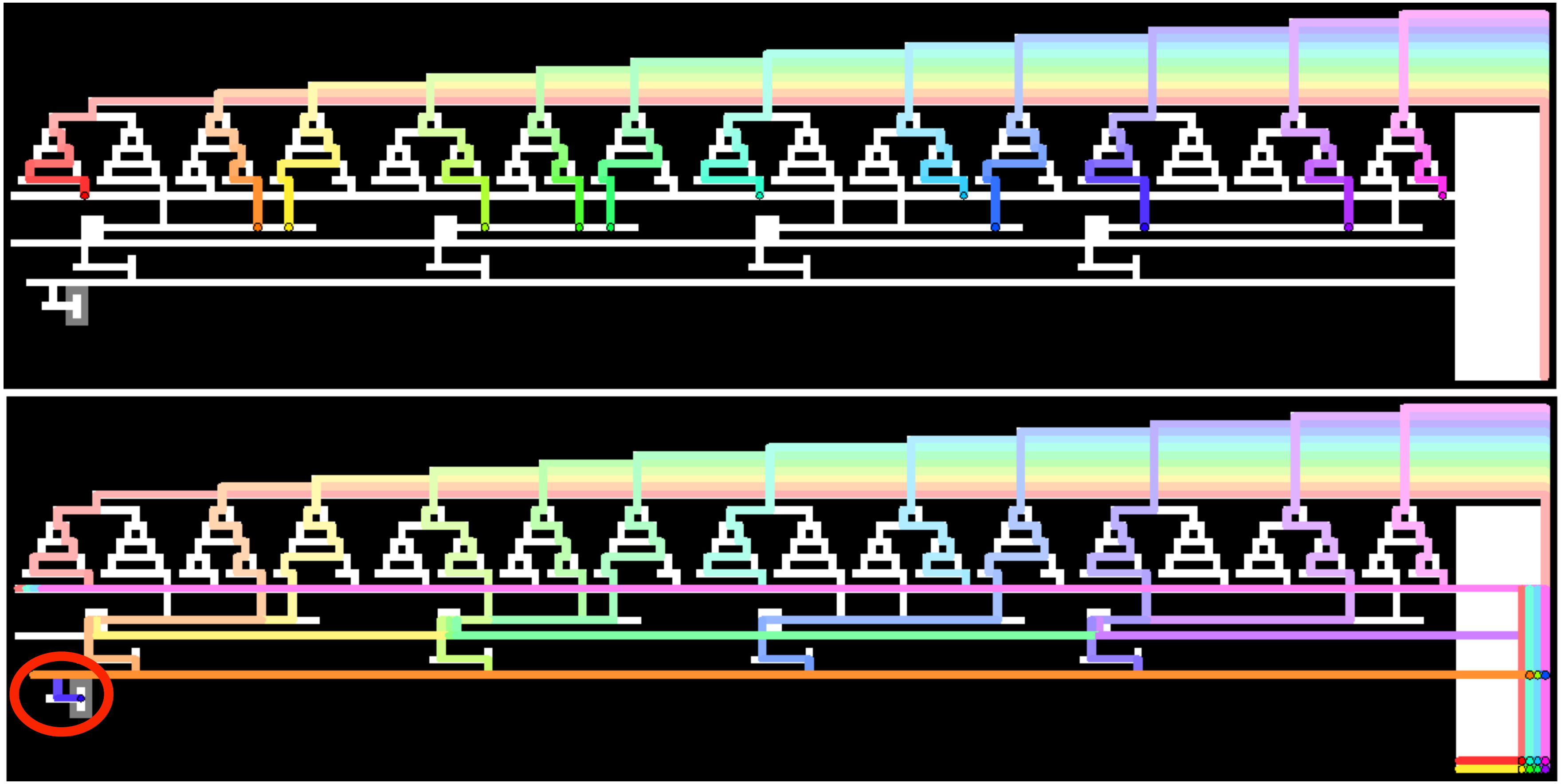
$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$



Complexity: Overall Construction

$(\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_3)$

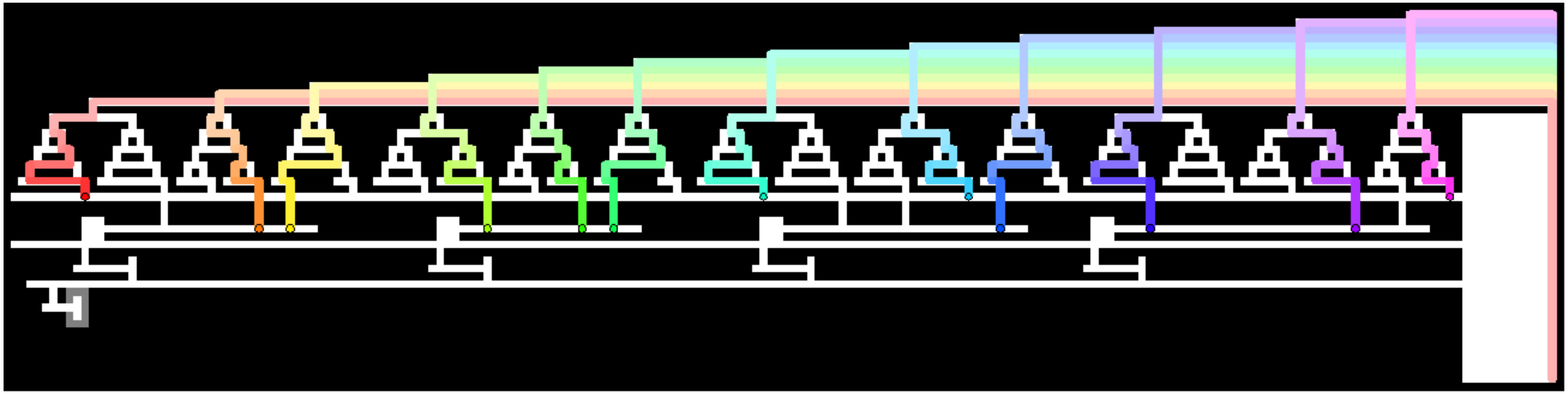
$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$

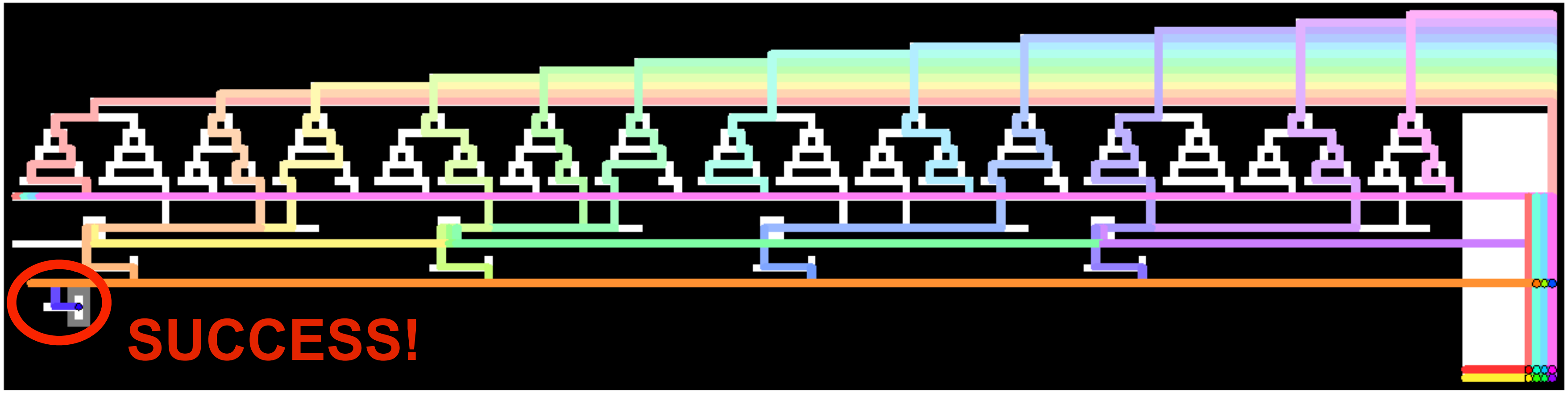


Complexity: Overall Construction

$(\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_3)$

$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$





Complexity: Summary

Complexity: Summary

Theorem 1. GLOBALCONTROL-MANYPARTICLES *is NP-hard: given an initial configuration of movable particles and fixed obstacles, it is NP-hard to decide whether any particle can be moved to a specified location.*

Part 4.2: Why Obstacles Are a Blessing

Life without Obstacles

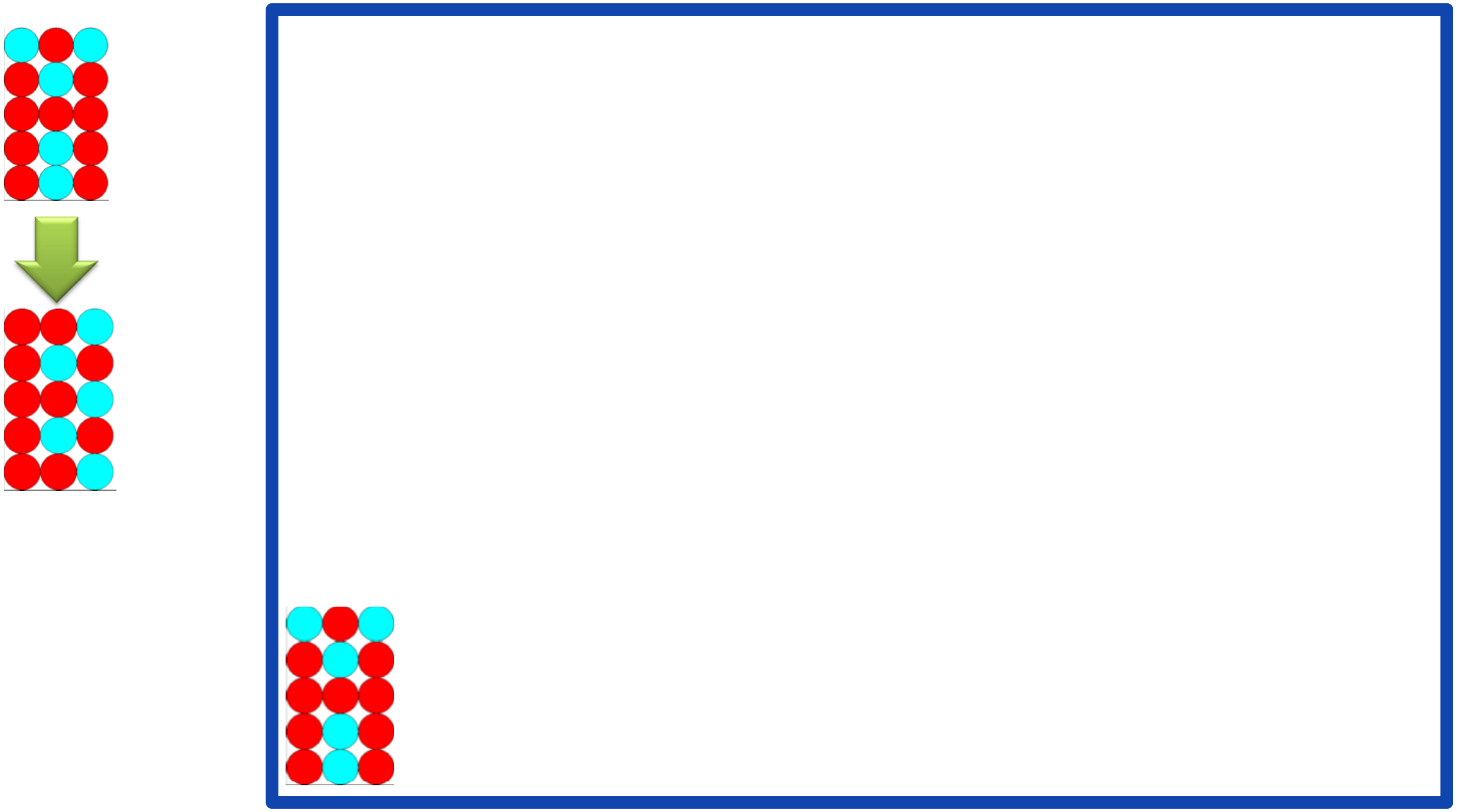
Life without Obstacles

Life without Obstacles

Lack of obstacles can be harmful!

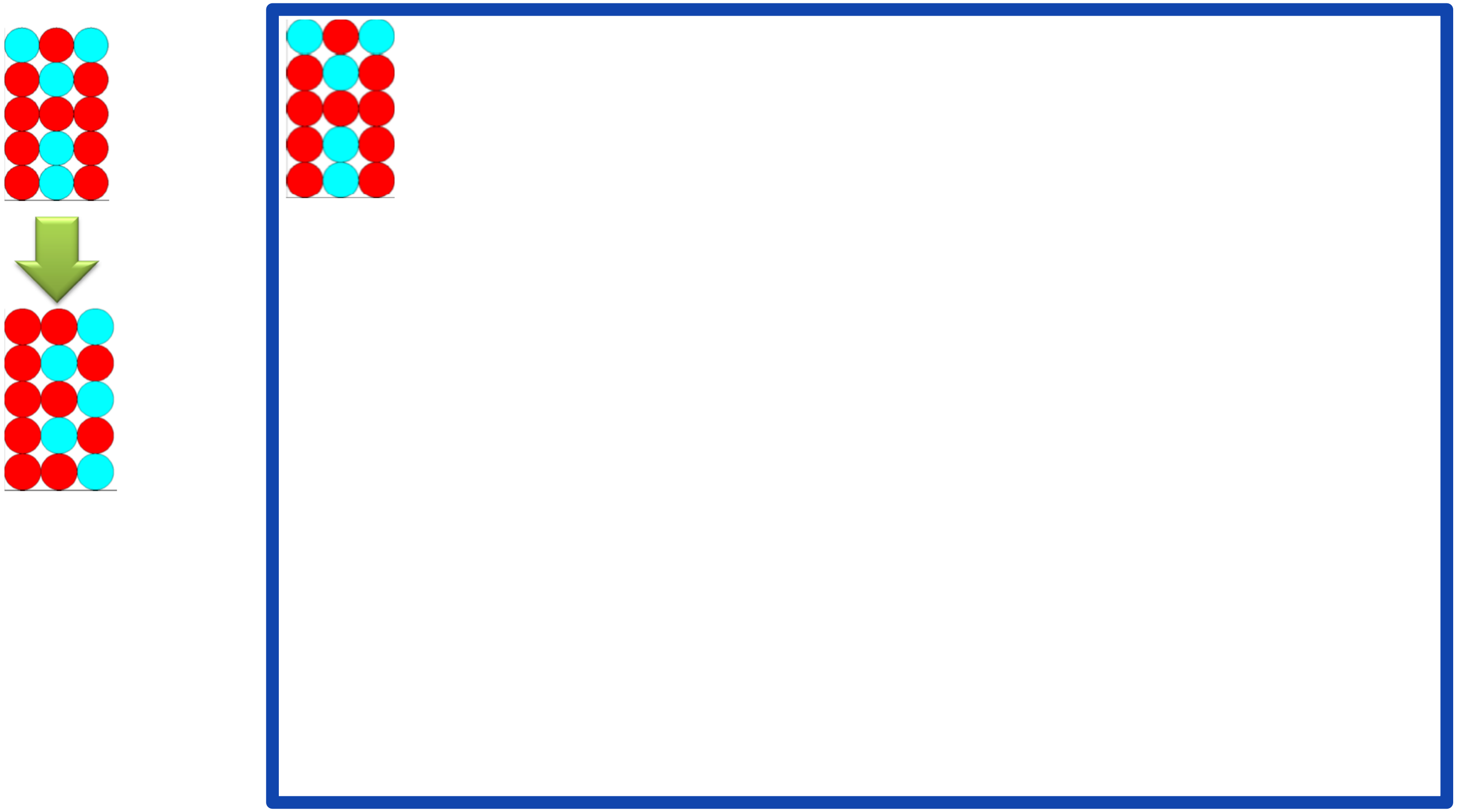
Life without Obstacles

Lack of obstacles can be harmful!



Life without Obstacles

Lack of obstacles can be harmful!



Life without Obstacles

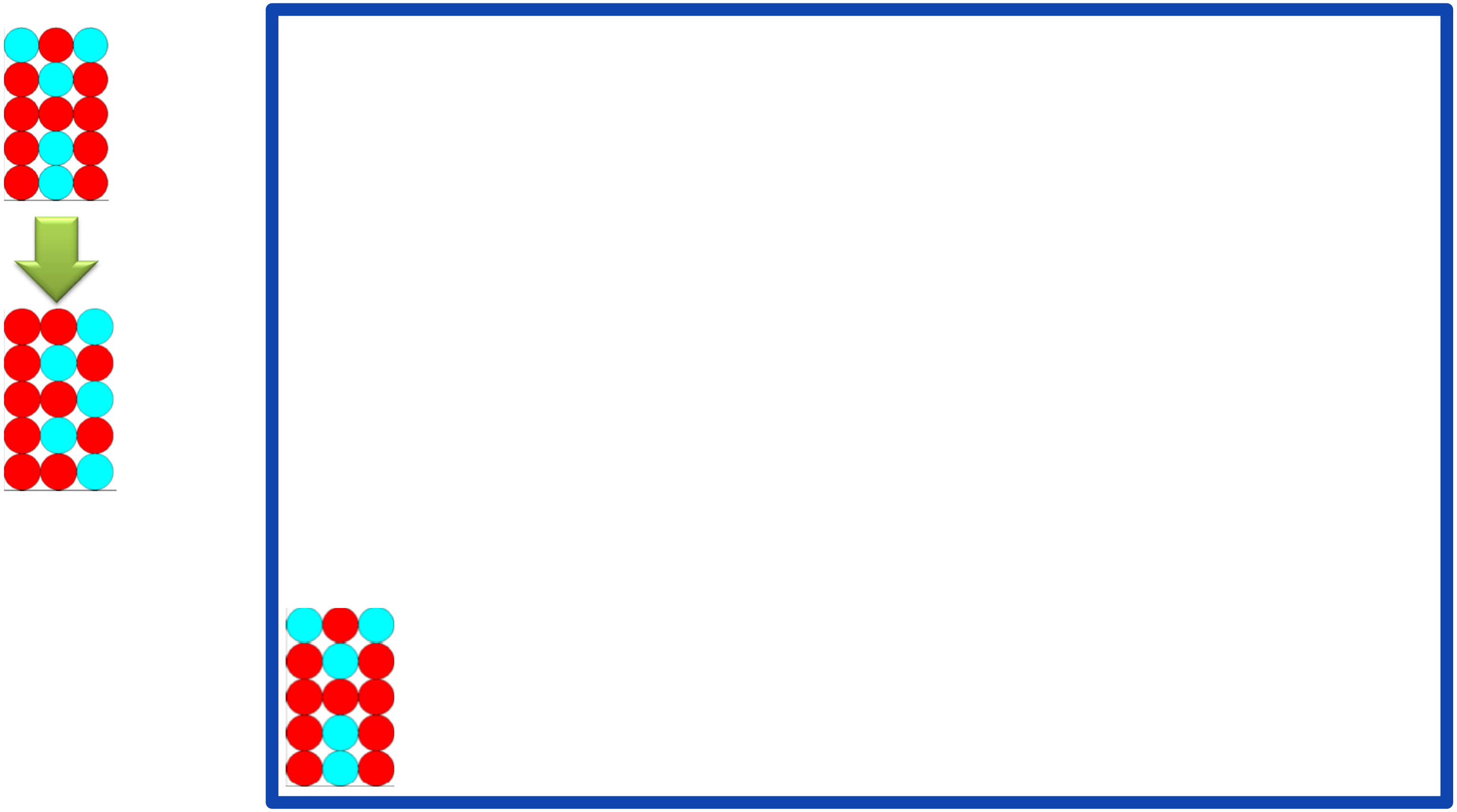
Lack of obstacles can be harmful!

Life without Obstacles

Lack of obstacles can be harmful!

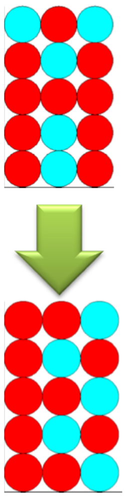
Life without Obstacles

Lack of obstacles can be harmful!



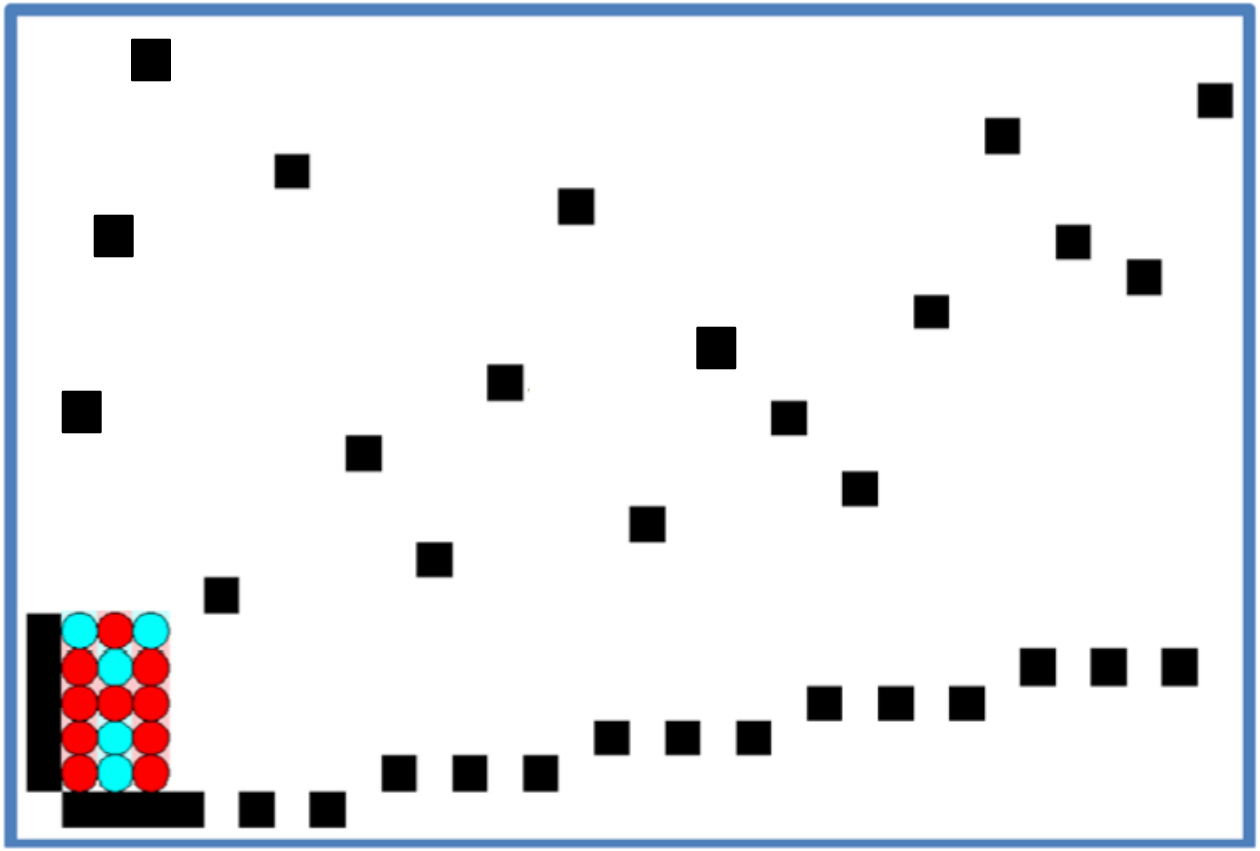
Life without Obstacles

Lack of obstacles can be harmful!

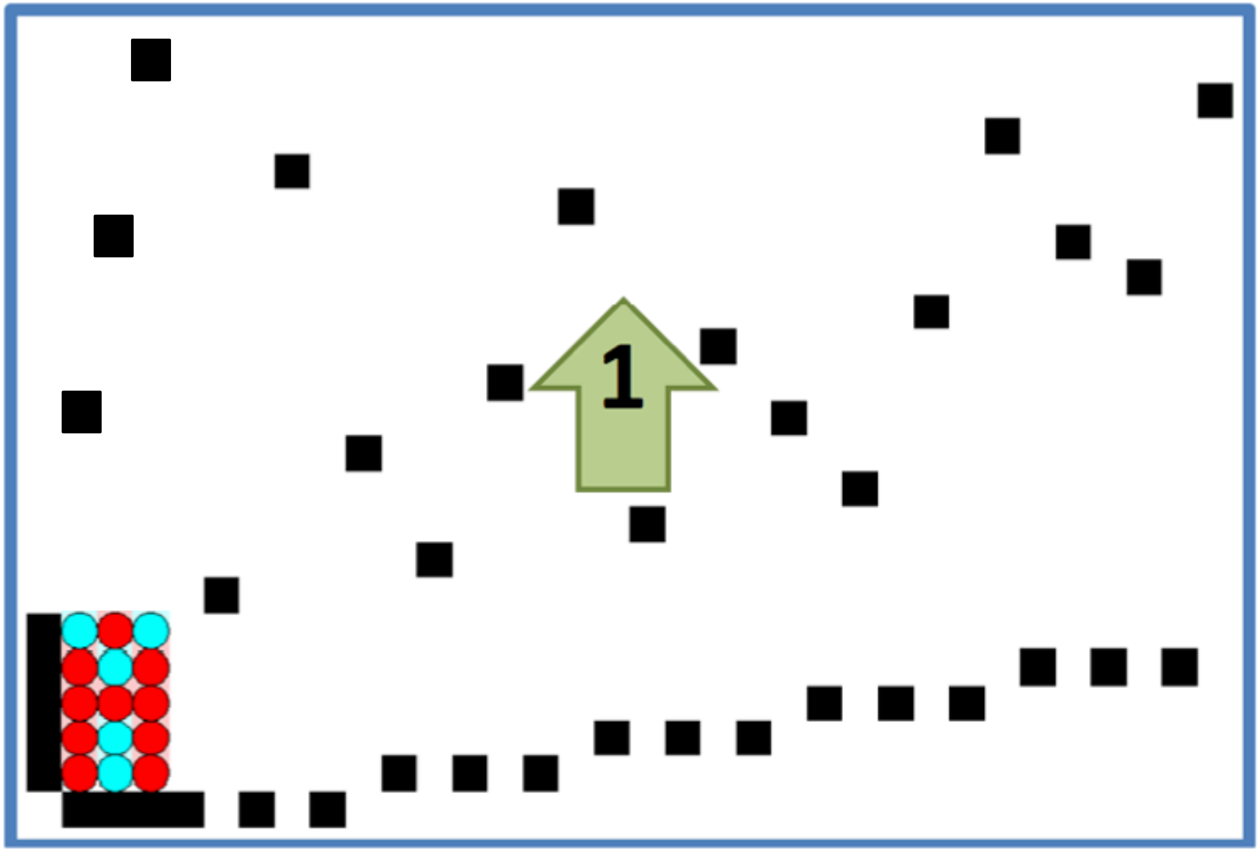


How Obstacles Can Be Helpful

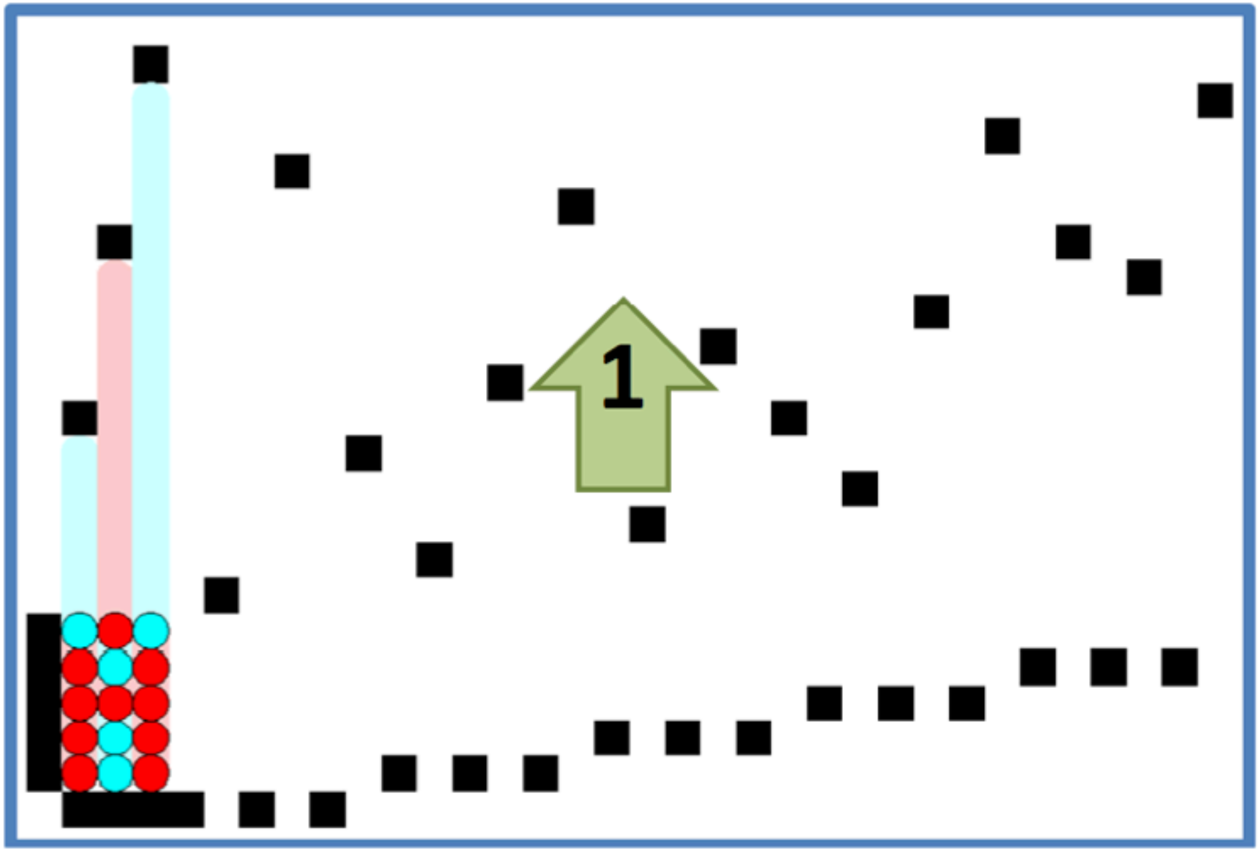
How Obstacles Can Be Helpful



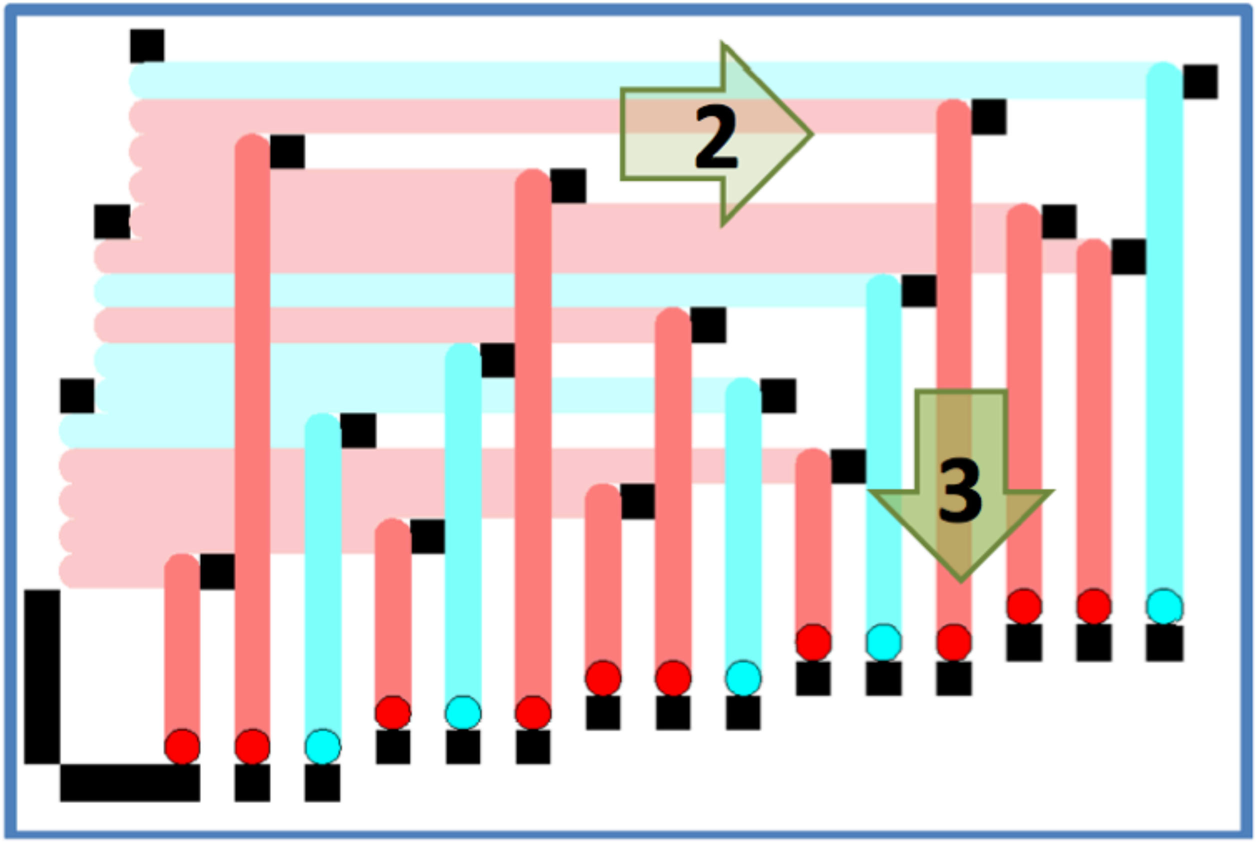
How Obstacles Can Be Helpful



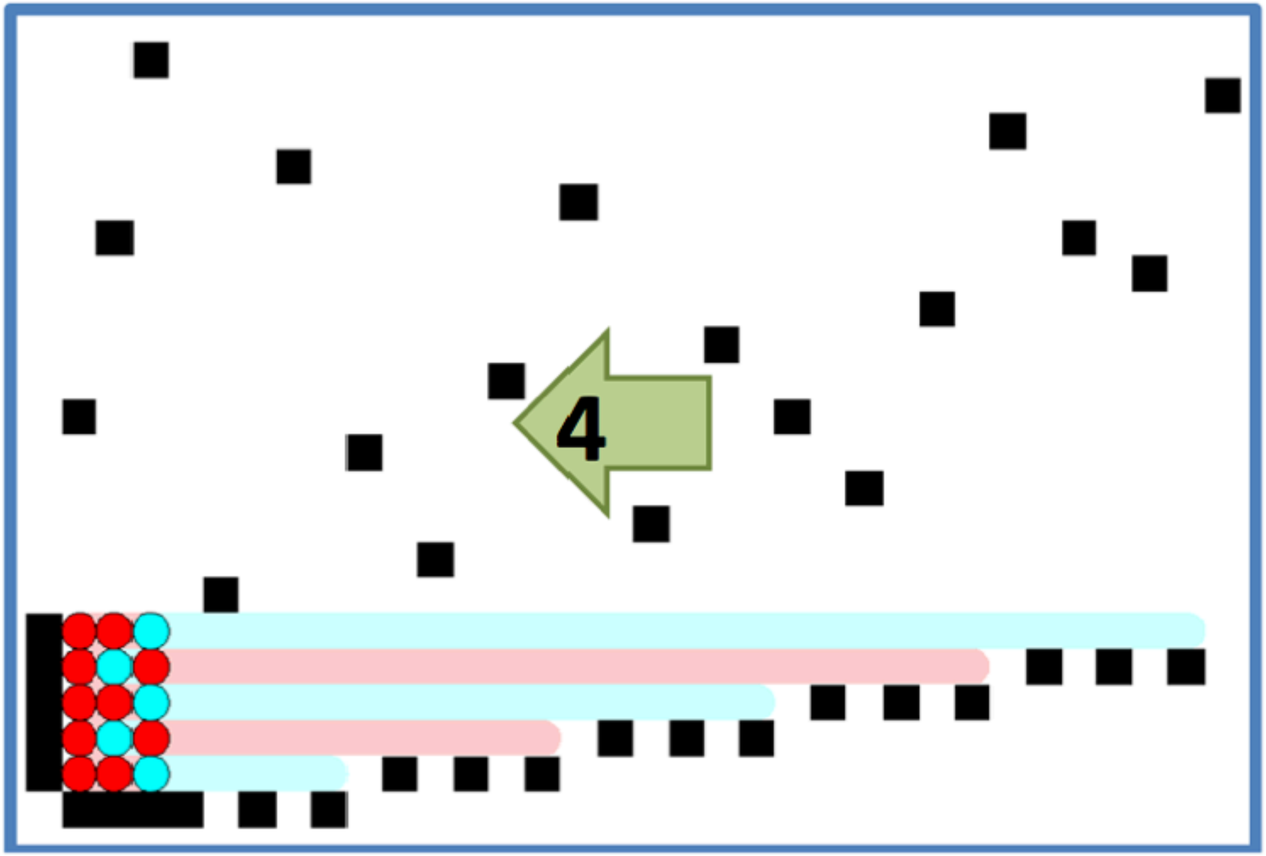
How Obstacles Can Be Helpful



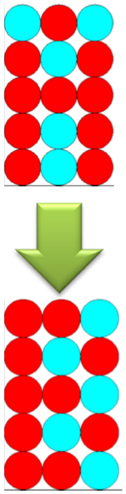
How Obstacles Can Be Helpful

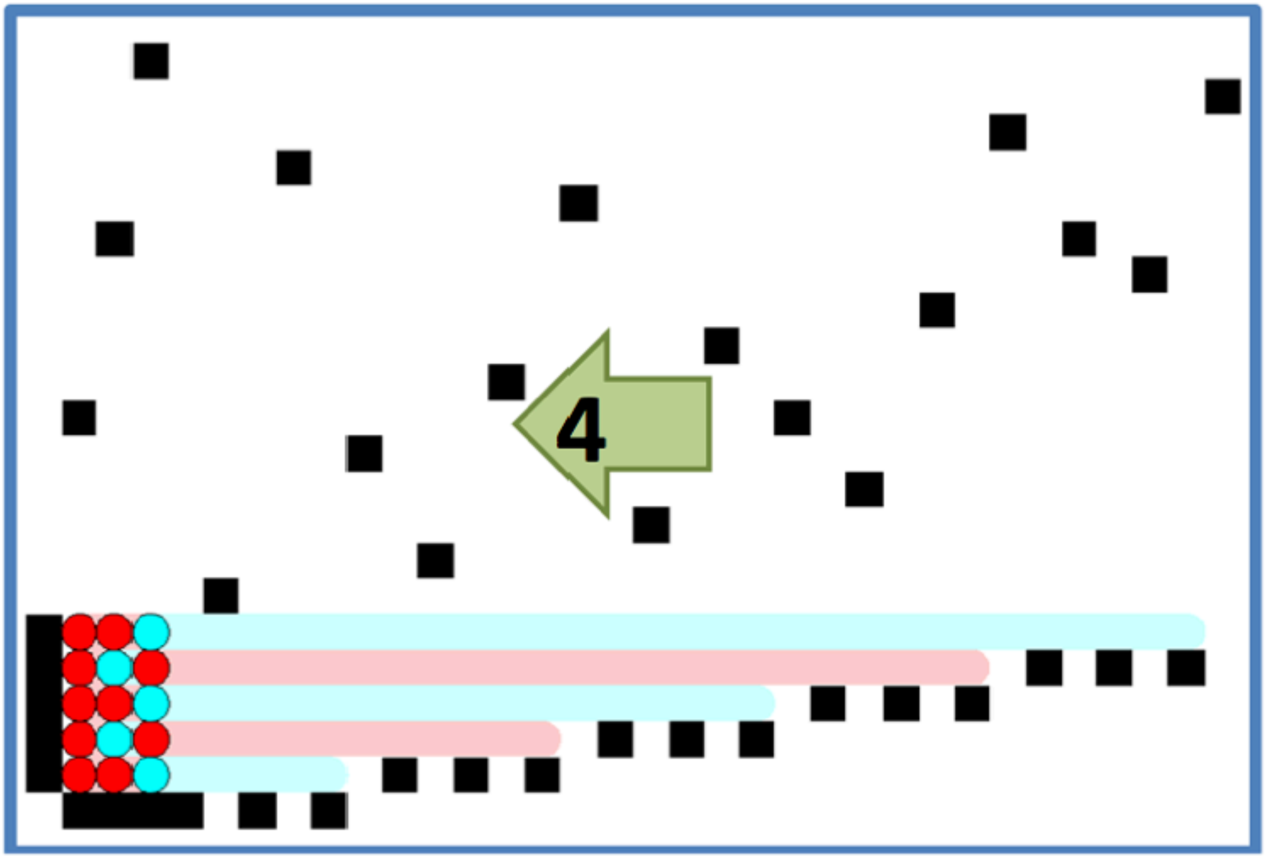


How Obstacles Can Be Helpful



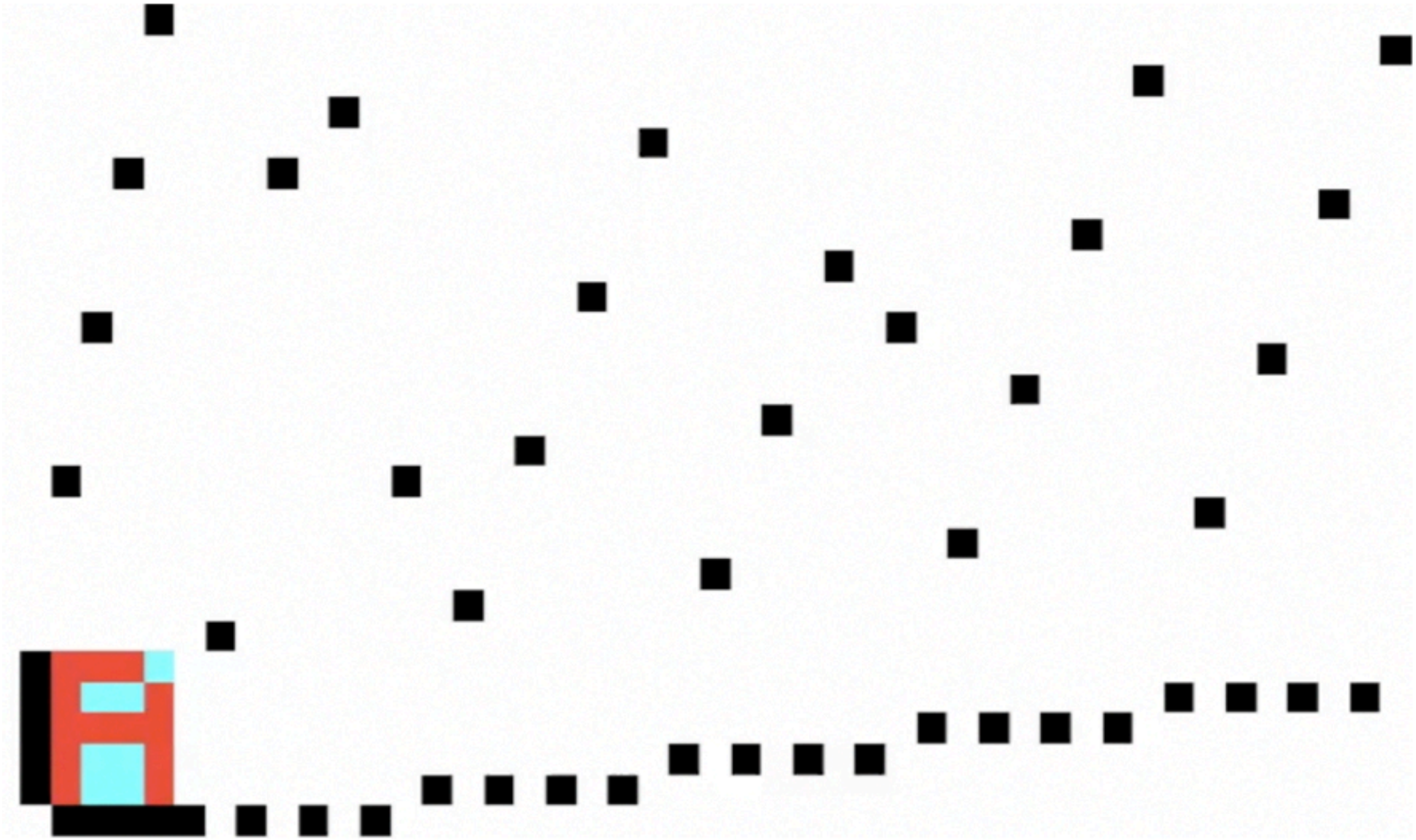
How Obstacles Can Be Helpful



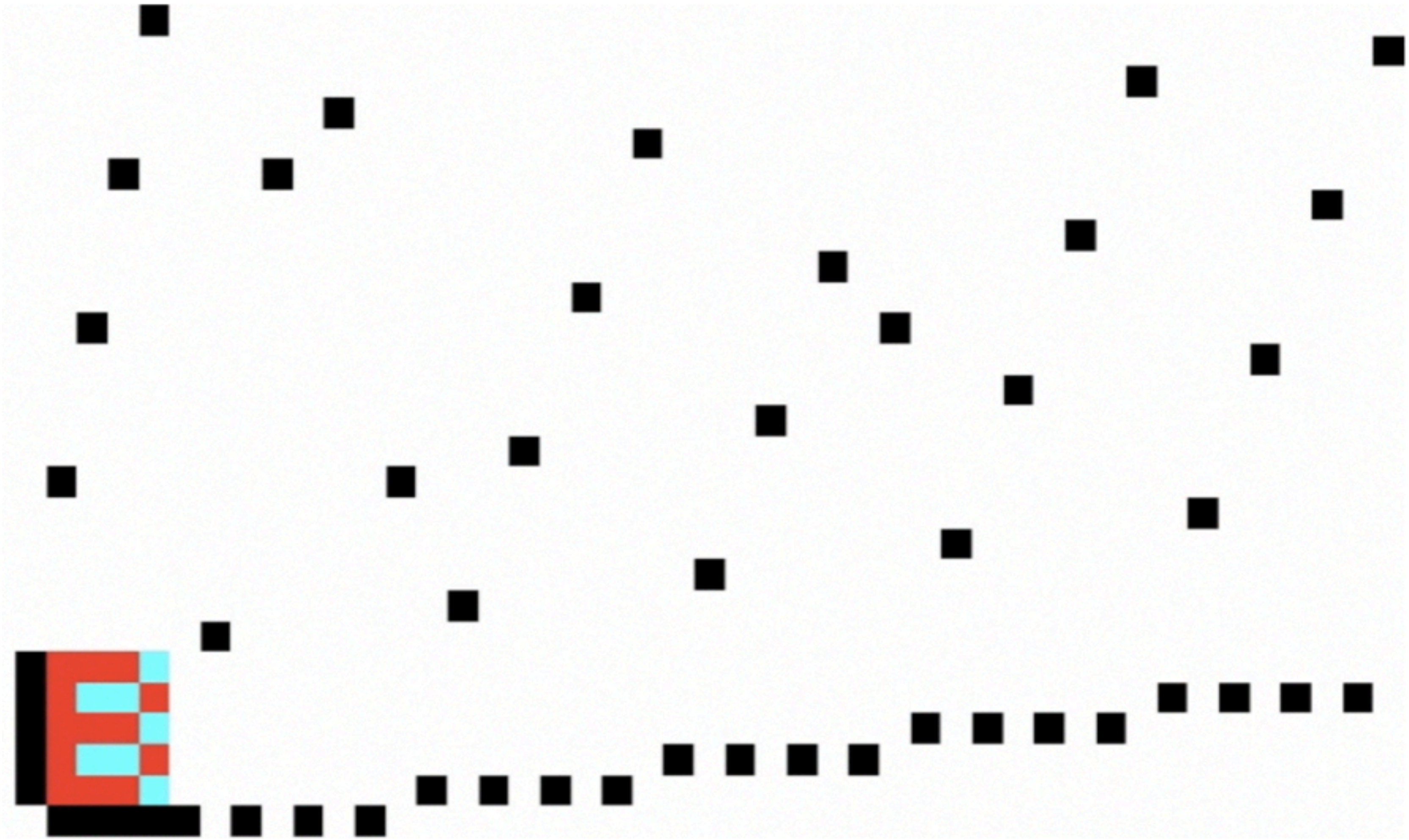


More Obstacle Action!

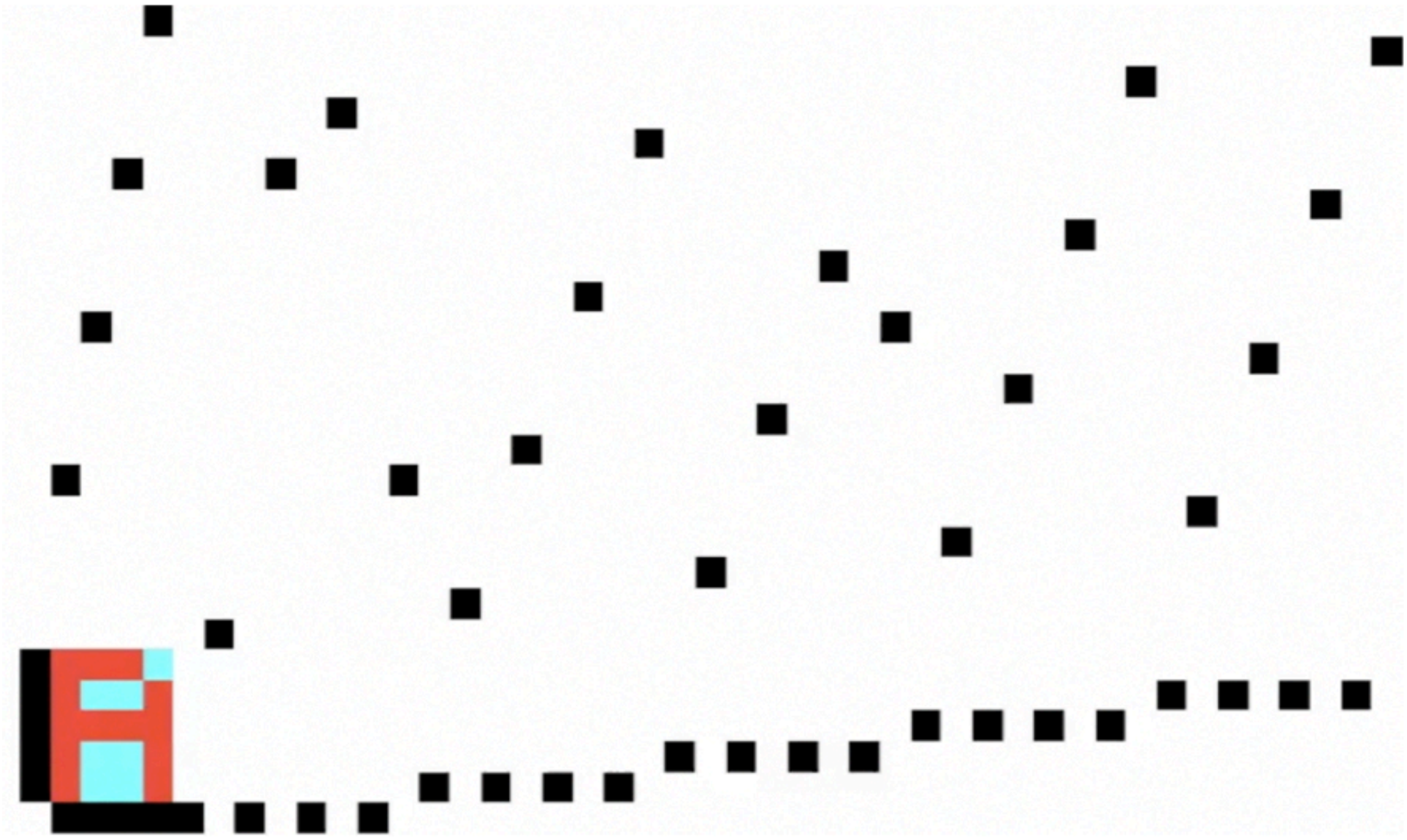
More Obstacle Action!



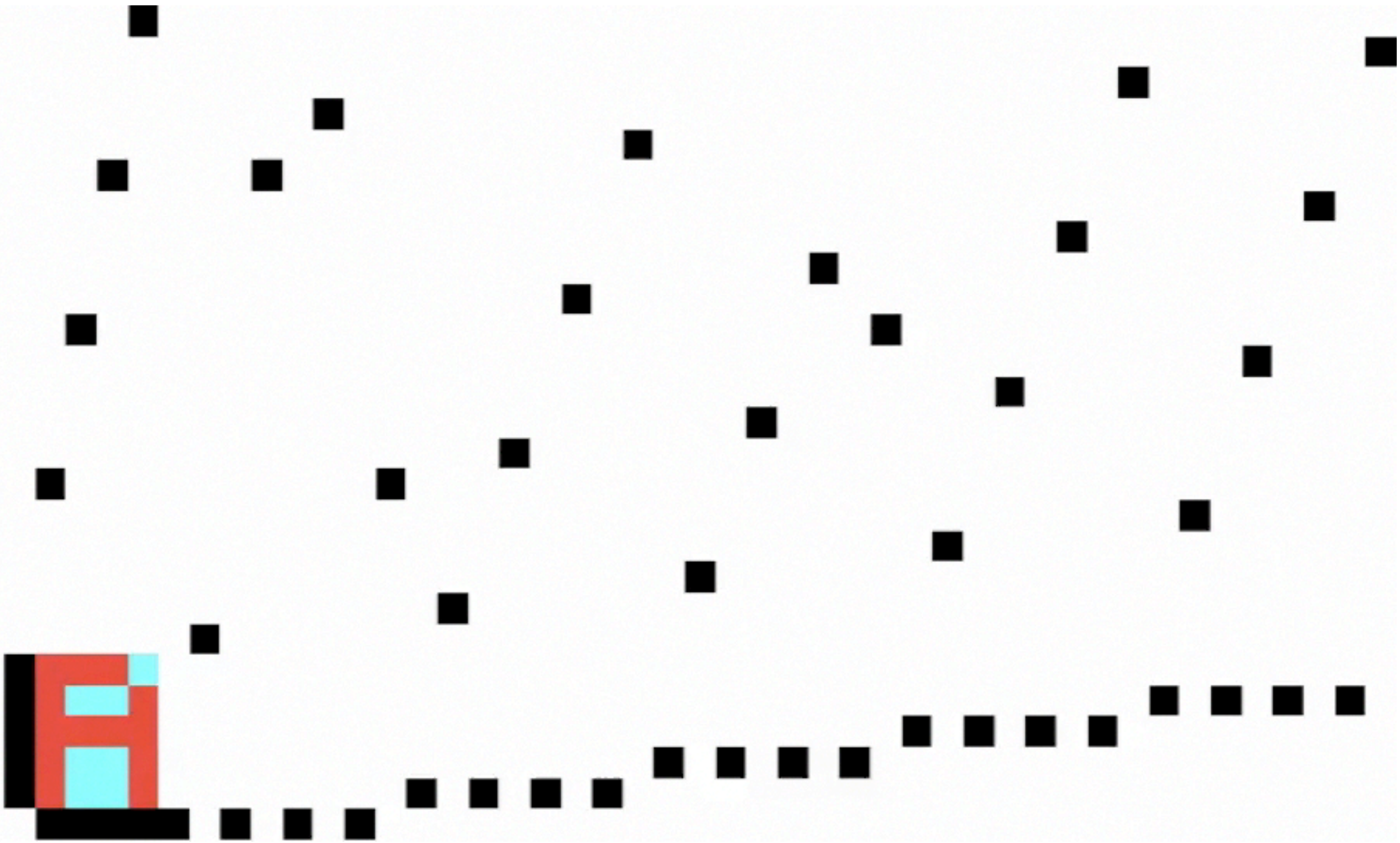
More Obstacle Action!



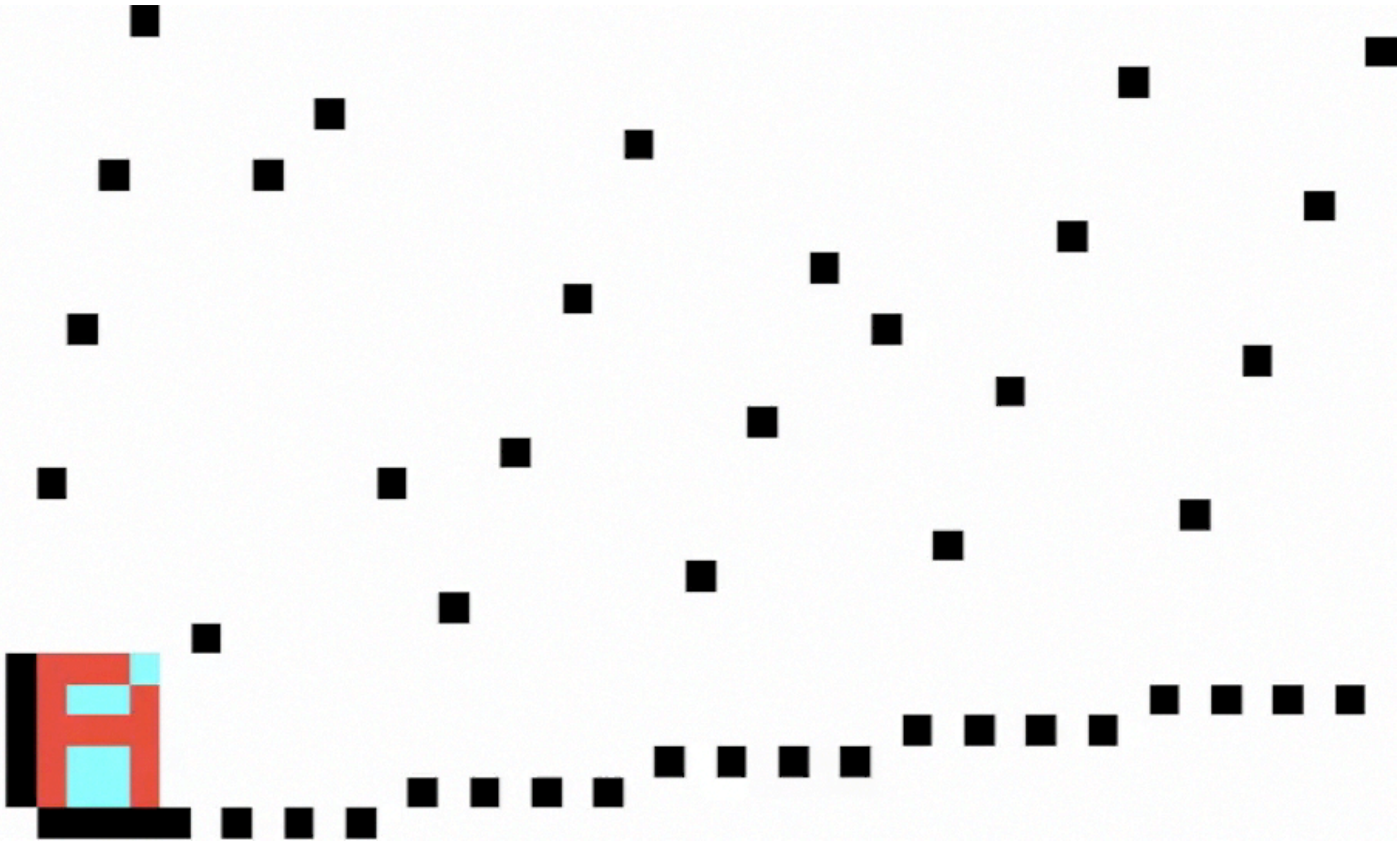
More Obstacle Action!



More Obstacle Action!



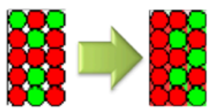
More Obstacle Action!

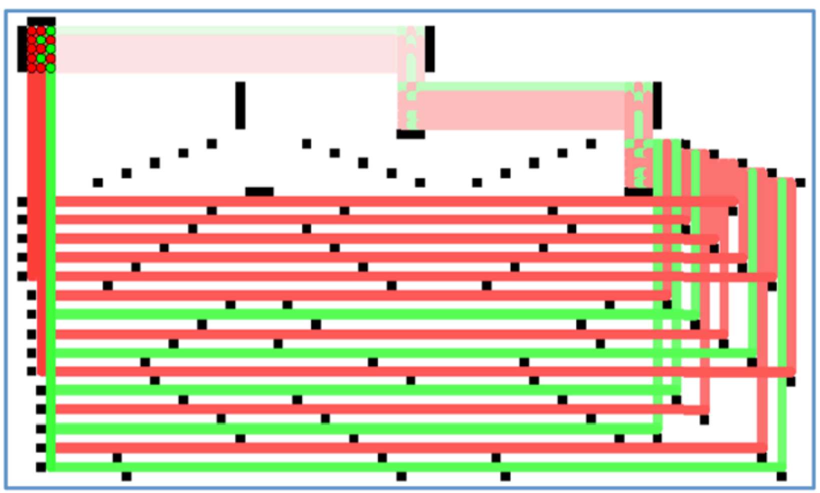


Multiple Permutations

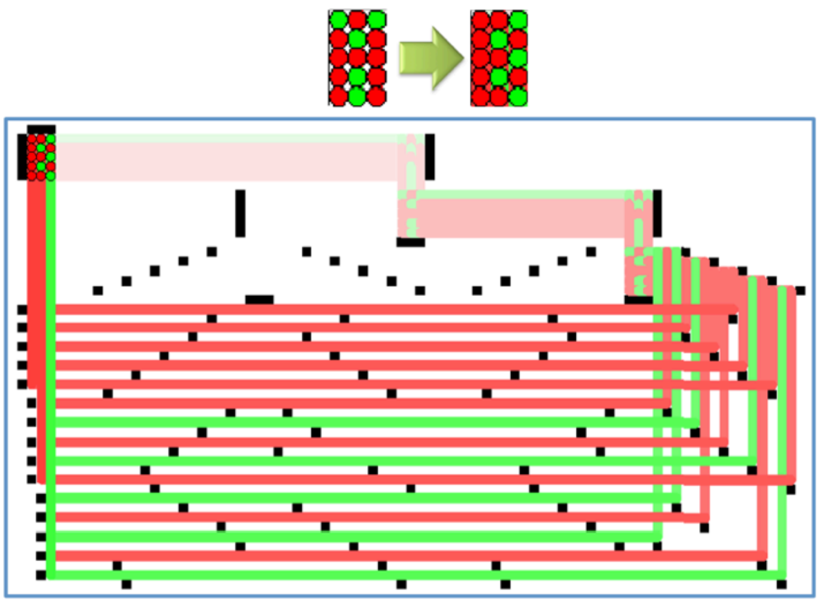
Multiple Permutations

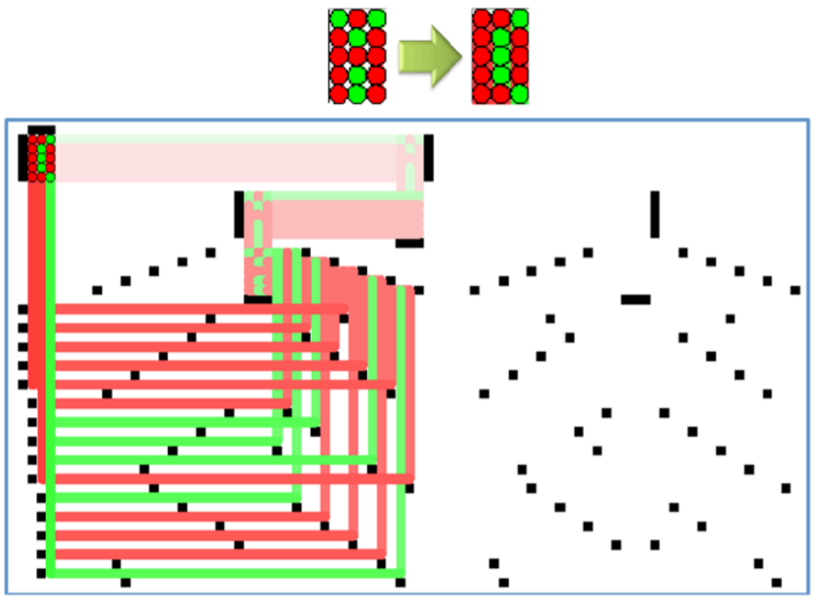
Multiple Permutations



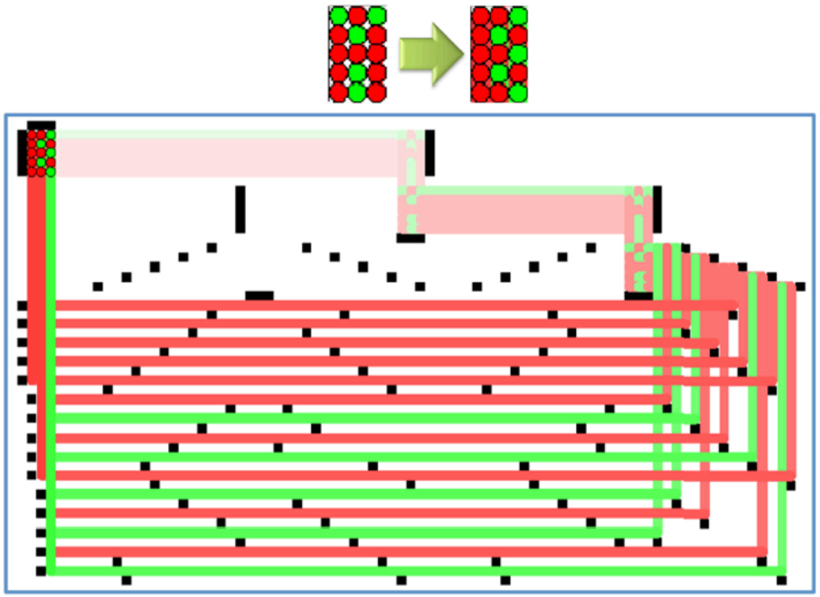


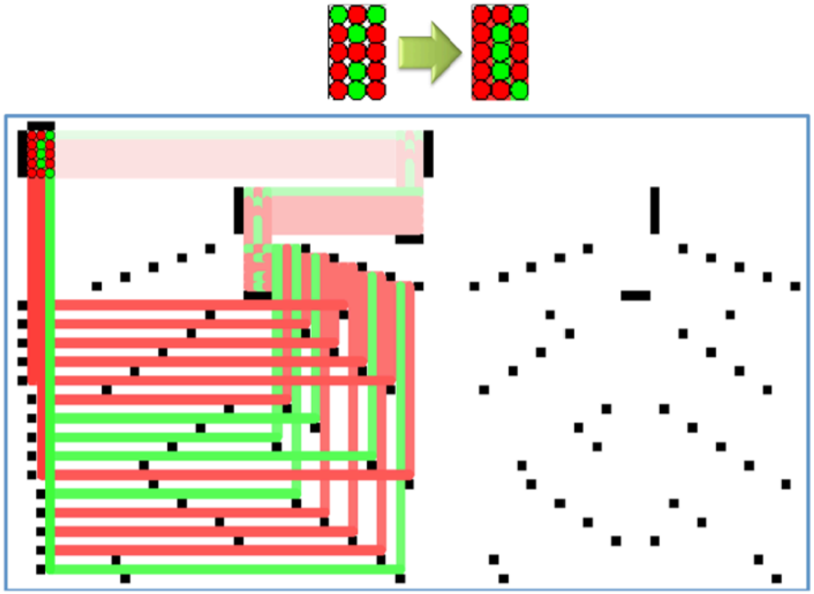
Multiple Permutations





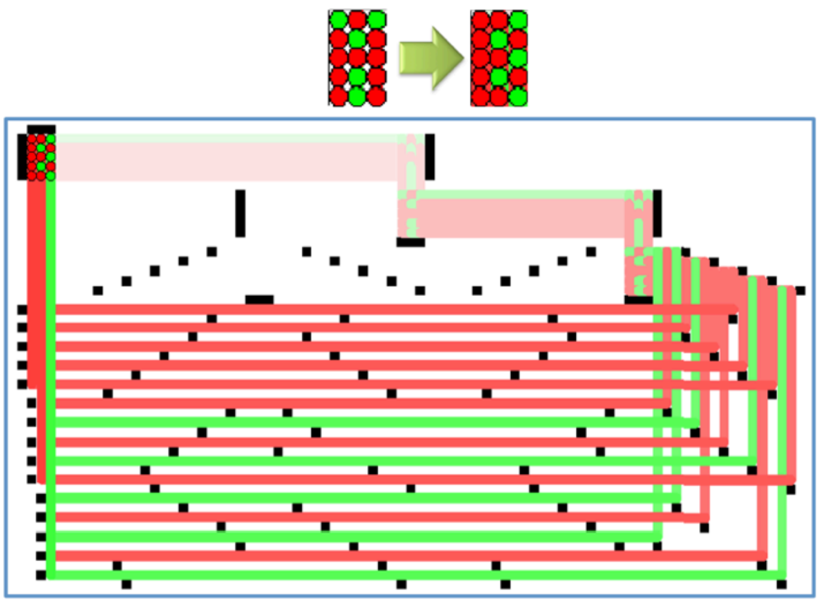
Multiple Permutations

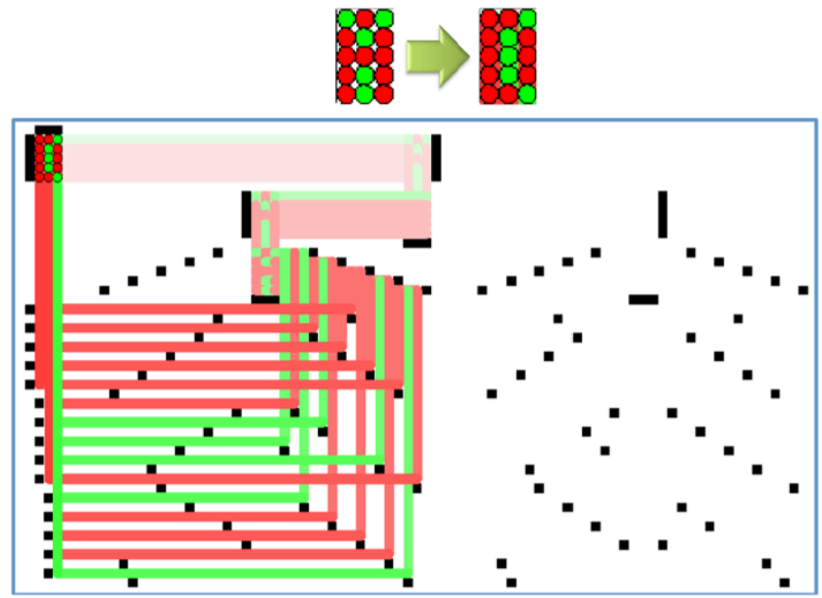




Multiple Permutations

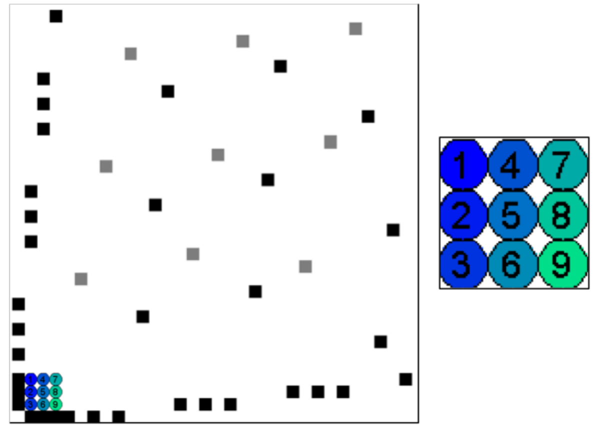
Theorem 3. *For any set of k fixed, but arbitrary, permutations of $n \times n$ pixels, we can construct a set of $O(kN)$ obstacles, such that we can switch from a start arrangement into any of the k permutations using at most $O(\log k)$ force-field moves.*



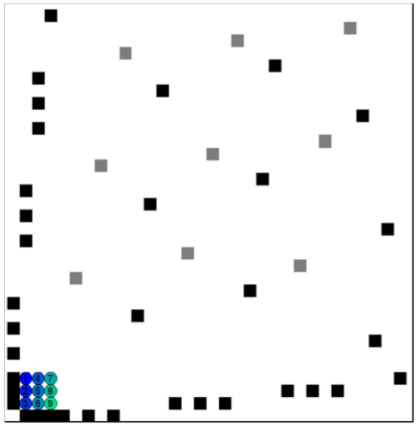


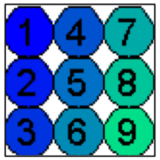
Designing Obstacles

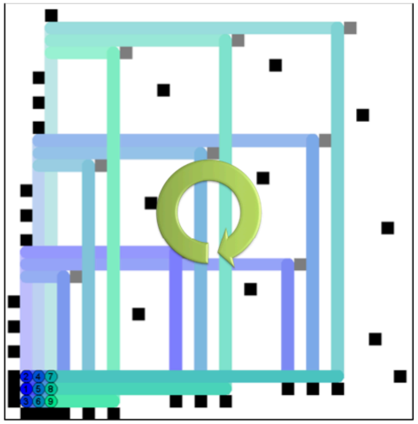
Designing Obstacles



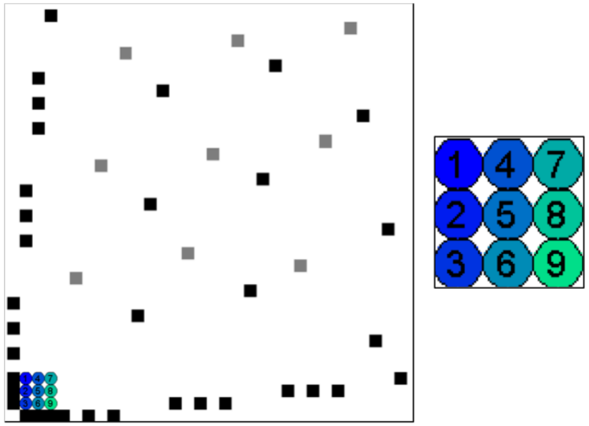
Designing Obstacles

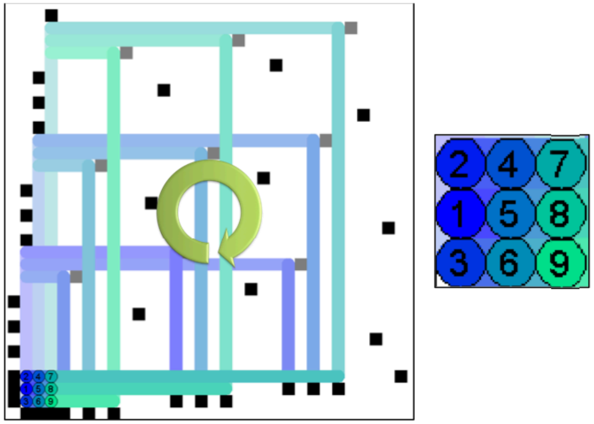






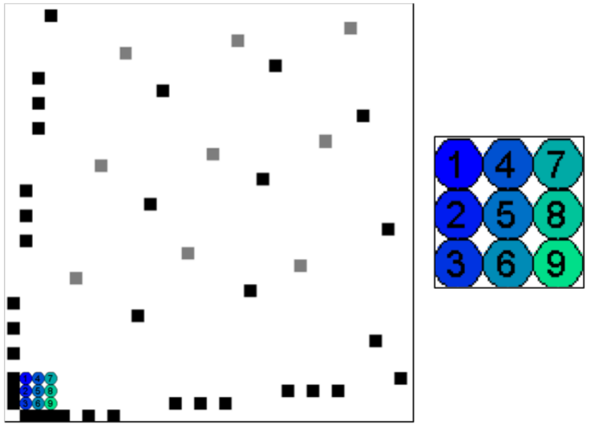
Designing Obstacles

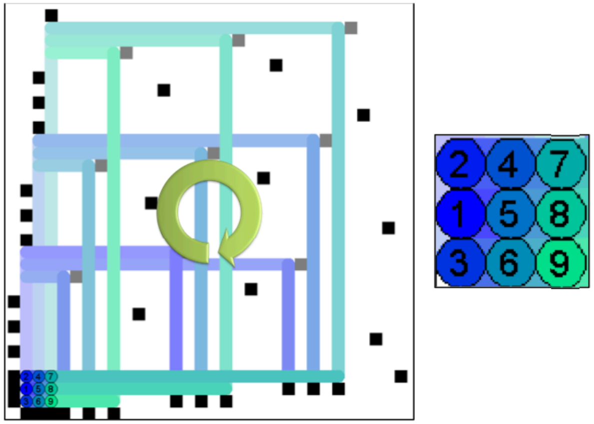


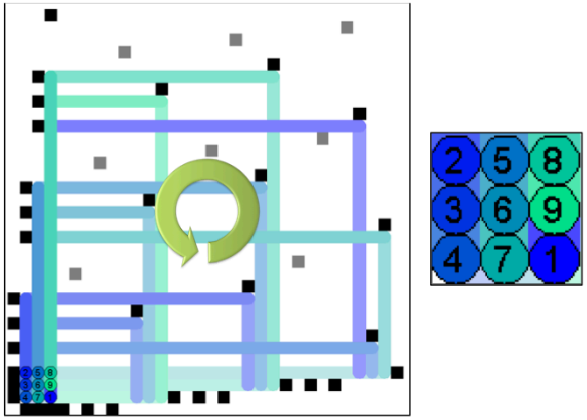


CW: (12)

Designing Obstacles

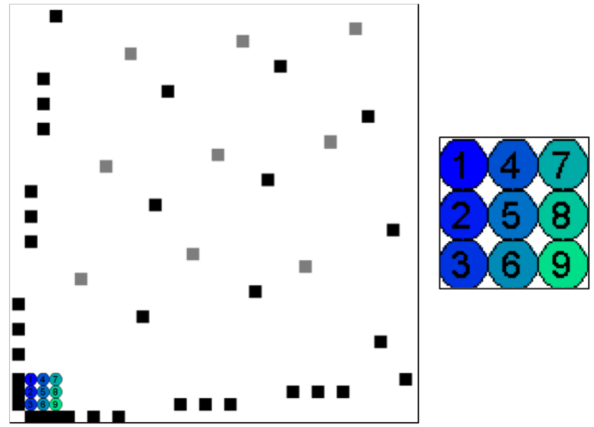


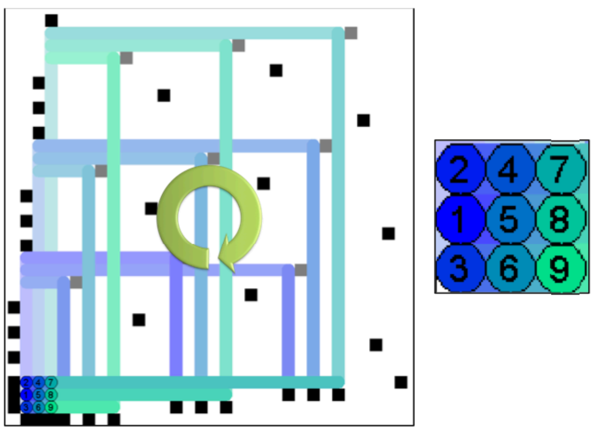


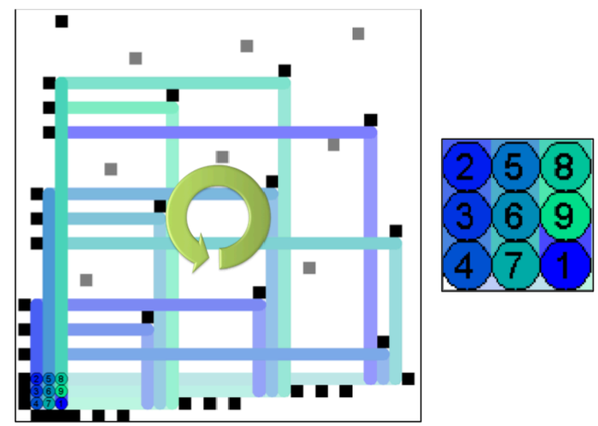


CW: (12)

Designing Obstacles



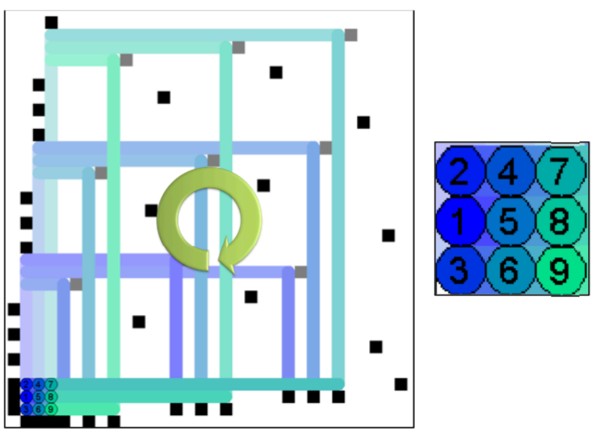




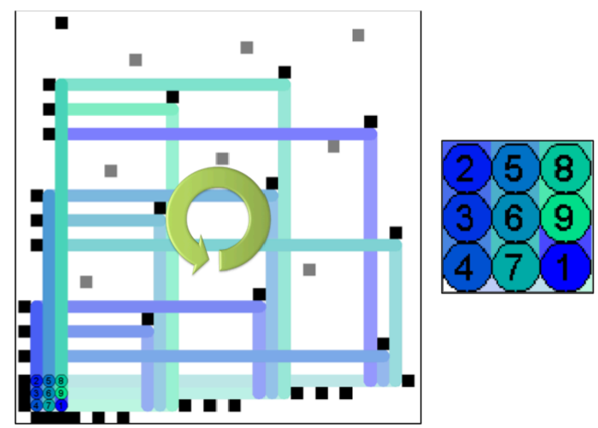
CW: (12)

CCW: (123456789)

Designing Obstacles



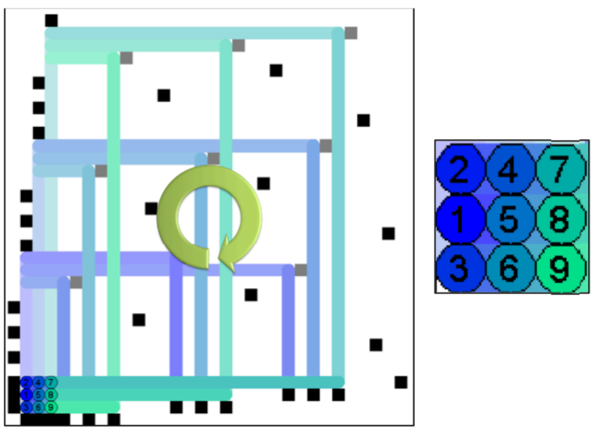
CW: (12)



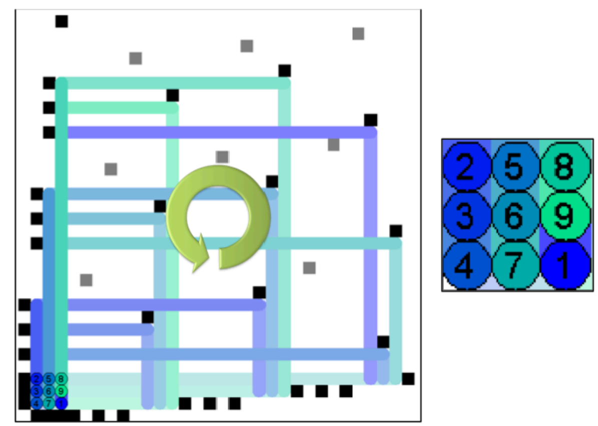
CCW: (123456789)

Designing Obstacles

Lemma 5. *Any permutation of N objects can be generated by the two base permutations $p = (12)$ and $q = (12 \cdots N)$. Moreover, any permutation can be generated by a sequence of length at most N^2 that consists of p and q.*



CW: (12)

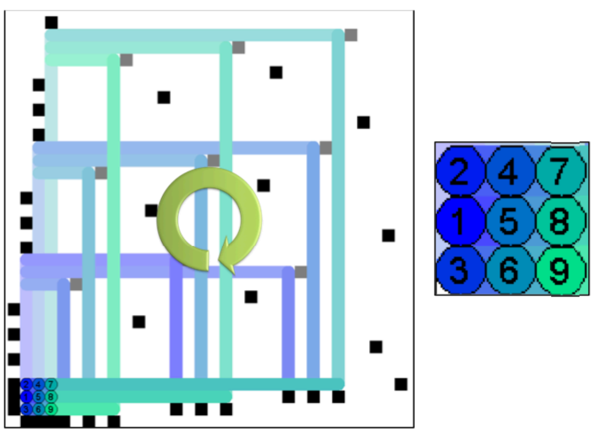


CCW: (123456789)

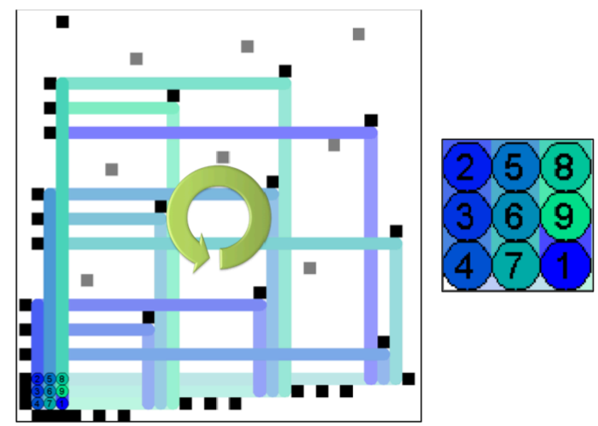
Designing Obstacles

Lemma 5. *Any permutation of N objects can be generated by the two base permutations $p = (12)$ and $q = (12 \cdots N)$. Moreover, any permutation can be generated by a sequence of length at most N^2 that consists of p and q.*

Theorem 6. *We can construct a set of $O(N)$ obstacles such that any $n \times n$ arrangement of N pixels can be rearranged into any other $n \times n$ arrangement π of the same pixels, using at most $O(N^2)$ force-field moves.*



CW: (12)



CCW: (123456789)

Designing Obstacles

Designing Obstacles

Lemma 7. *Any permutation of N objects can be generated by the N base permutations $p_1 = (12), p_2 = (13), \ldots, p_{N-1} = (1(N-1))$ and $q = (12 \cdots N)$. Moreover, any permutation can be generated by a sequence of length at most N that consists of the p_i and q.*

Designing Obstacles

Lemma 7. *Any permutation of N objects can be generated by the N base permutations $p_1 = (12), p_2 = (13), \ldots, p_{N-1} = (1(N-1))$ and $q = (12\cdots N)$. Moreover, any permutation can be generated by a sequence of length at most N that consists of the p_i and q.*

Theorem 8. *We can construct a set of $O(N^2)$ obstacles such that any $n \times n$ arrangement of N pixels can be rearranged into any other $n \times n$ arrangement π of the same pixels, using at most $O(N \log N)$ force-field moves.*

Designing Obstacles

Lemma 7. *Any permutation of N objects can be generated by the N base permutations $p_1 = (12), p_2 = (13), \ldots, p_{N-1} = (1(N-1))$ and $q = (12 \cdots N)$. Moreover, any permutation can be generated by a sequence of length at most N that consists of the p_i and q.*

Theorem 8. *We can construct a set of $O(N^2)$ obstacles such that any $n \times n$ arrangement of N pixels can be rearranged into any other $n \times n$ arrangement π of the same pixels, using at most $O(N \log N)$ force-field moves.*

Theorem 9. *Suppose we have a set of obstacles such that any permutation of an $n \times n$ arrangement of pixels can be achieved by at most M force-field moves. Then M is at least $\Omega(N \log N)$.*

Proof. Each permutation must be achieved by a sequence of force-field moves. Because each decision for a force-field move $\{u, d, l, r\}$ partitions the remaining set of possible permutations into at most four different subsets, we need at least $\Omega(\log(N!)) = \Omega(N \log N)$ such moves. □

Breaking News:
PSPACE-Completeness!

More on Complexity!

More on Complexity!

Theoretical Computer Science 36 (1985) 265–289
North-Holland

THE COMPLEXITY OF FINDING MINIMUM-LENGTH GENERATOR SEQUENCES

Mark R. JERRUM

Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, Scotland (United Kingdom)

Communicated by M.S. Paterson
Received July 1983
Revised May 1984

Abstract. The computational complexity of the following problem is investigated: Given a permutation group specified as a set of generators, and a single target permutation which is a member of the group, what is the shortest expression for the target permutation in terms of the generators? The general problem is demonstrated to be PSPACE-complete and, indeed, is shown to remain so even when the generator set is restricted to contain only two permutations. The restriction on generator set cardinality is the best possible, as the problem becomes soluble in polynomial time if the generator set contains only one permutation. An interesting feature of this problem is that it does not fall under the headings of 'two person games' or 'formal languages' which cover the great majority of known PSPACE-complete problems. Some restricted versions of the problem.

More on Complexity!

Theoretical Computer Science 36 (1985) 265–289
North-Holland

THE COMPLEXITY OF FINDING MINIMUM-LENGTH GENERATOR SEQUENCES

Mark R. JERRUM

Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, Scotland (United Kingdom)

Communicated by M.S. Paterson
Received July 1983
Revised May 1984

Abstract. The computational complexity of the following problem is investigated: Given a permutation group specified as a set of generators, and a single target permutation which is a member of the group, what is the shortest expression for the target permutation in terms of the generators? The general problem is demonstrated to be PSPACE-complete and, indeed, is shown to remain so even when the generator set is restricted to contain only two permutations. The restriction on generator set cardinality is the best possible, as the problem becomes soluble in polynomial time if the generator set contains only one permutation. An interesting feature of this problem is that it does not fall under the headings of 'two person games' or 'formal languages' which cover the great majority of known PSPACE-complete problems. Some restricted versions of the problem.

More on Complexity!

Theoretical Computer Science 36 (1985) 265–289
North-Holland

THE COMPLEXITY OF FINDING MINIMUM-LENGTH GENERATOR SEQUENCES

Mark R. JERRUM

Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, Scotland (United Kingdom)

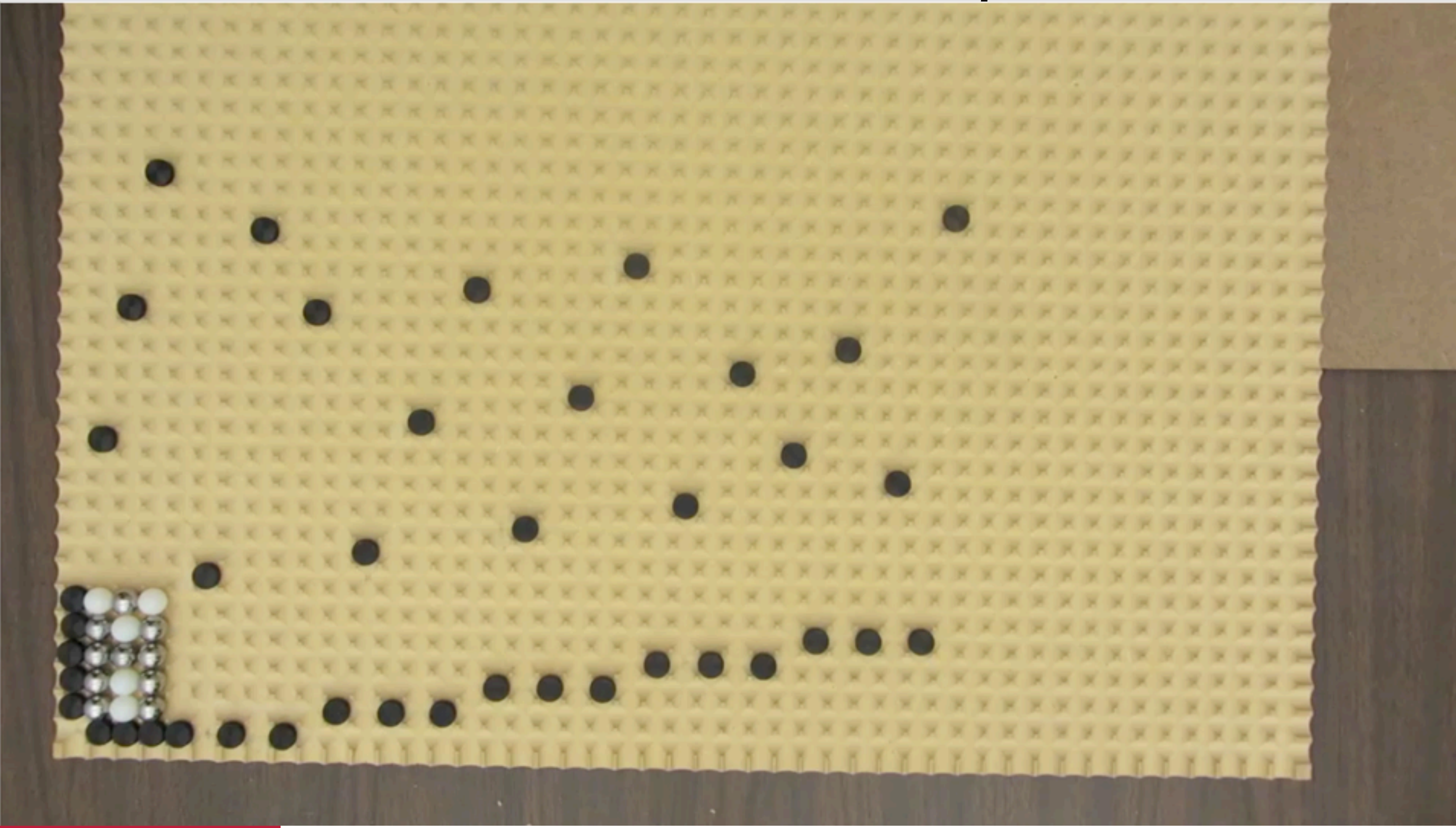
Communicated by M.S. Paterson
Received July 1983
Revised May 1984

Abstract. The computational complexity of the following problem is investigated: Given a permutation group specified as a set of generators, and a single target permutation which is a member of the group, what is the shortest expression for the target permutation in terms of the generators? The general problem is demonstrated to be PSPACE-complete and, indeed, is shown to remain so even when the generator set is restricted to contain only two permutations. The restriction on generator set cardinality is the best possible, as the problem becomes soluble in polynomial time if the generator set contains only one permutation. An interesting feature of this problem is that it does not fall under the headings of 'two person games' or 'formal languages' which cover the great majority of known PSPACE-complete problems. Some restricted versions of the problem.

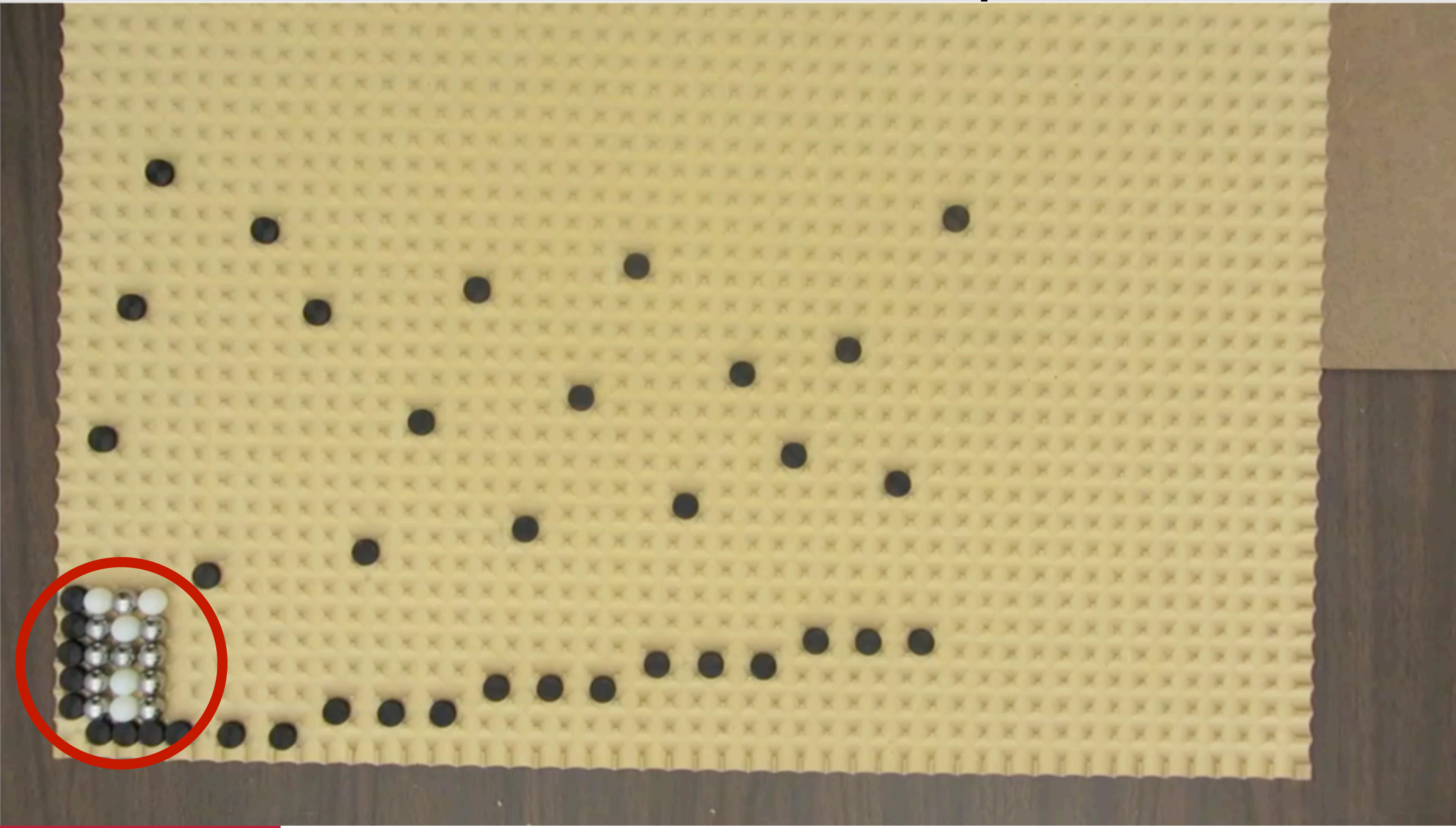
Part 4.3: A Real-World Demo!

Demo with Real Objects

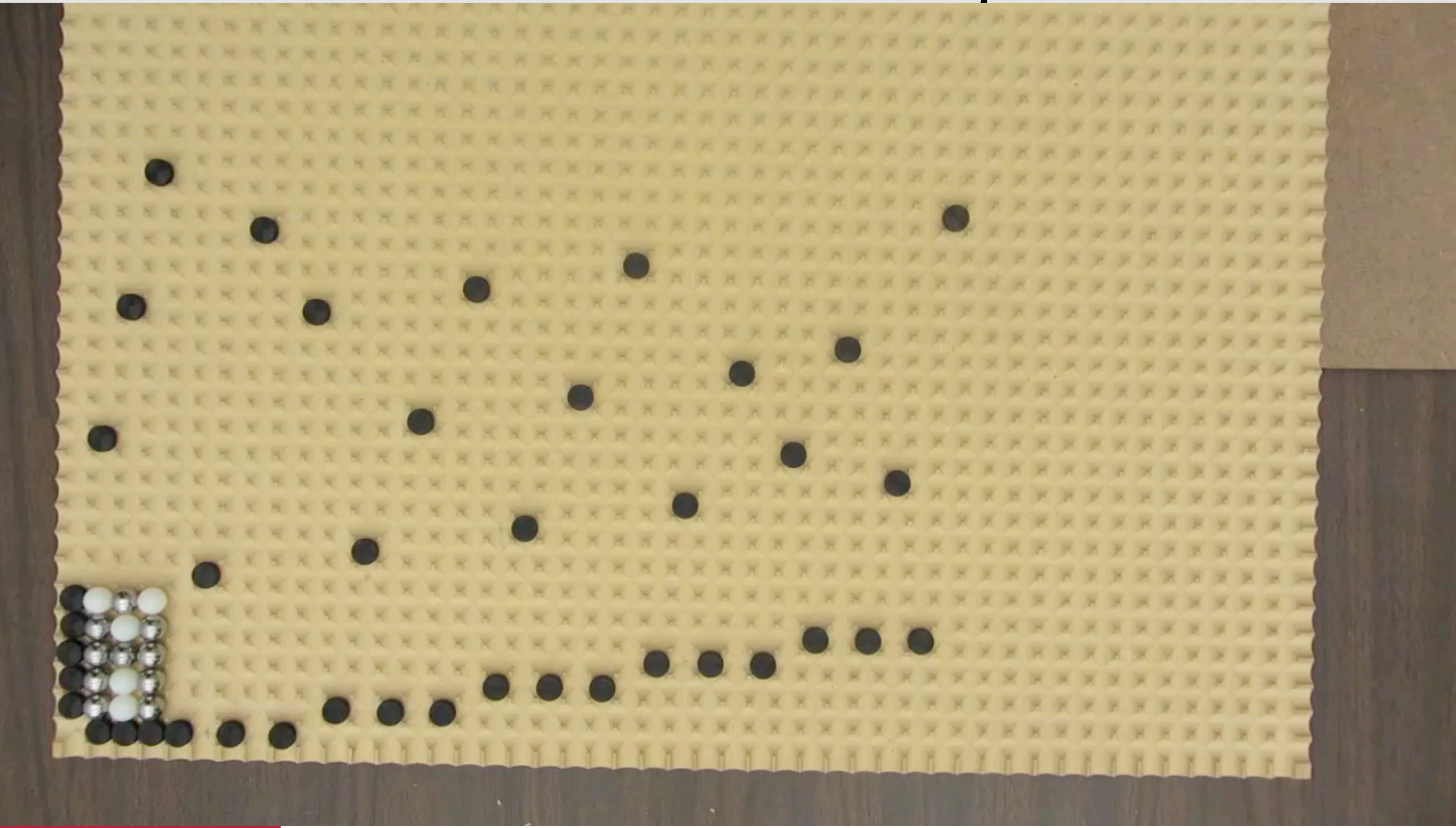
Demo with Real Objects



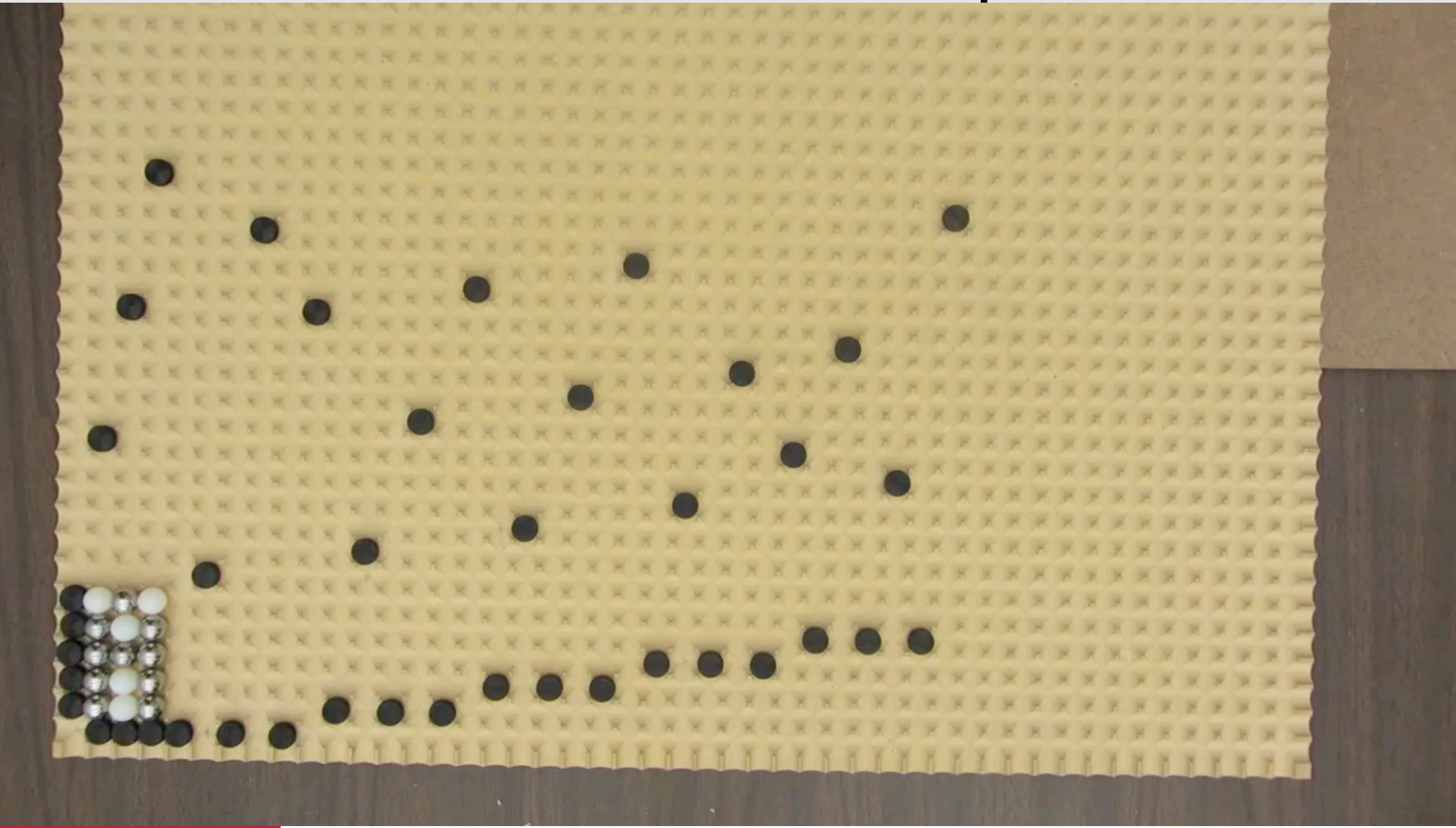
Demo with Real Objects



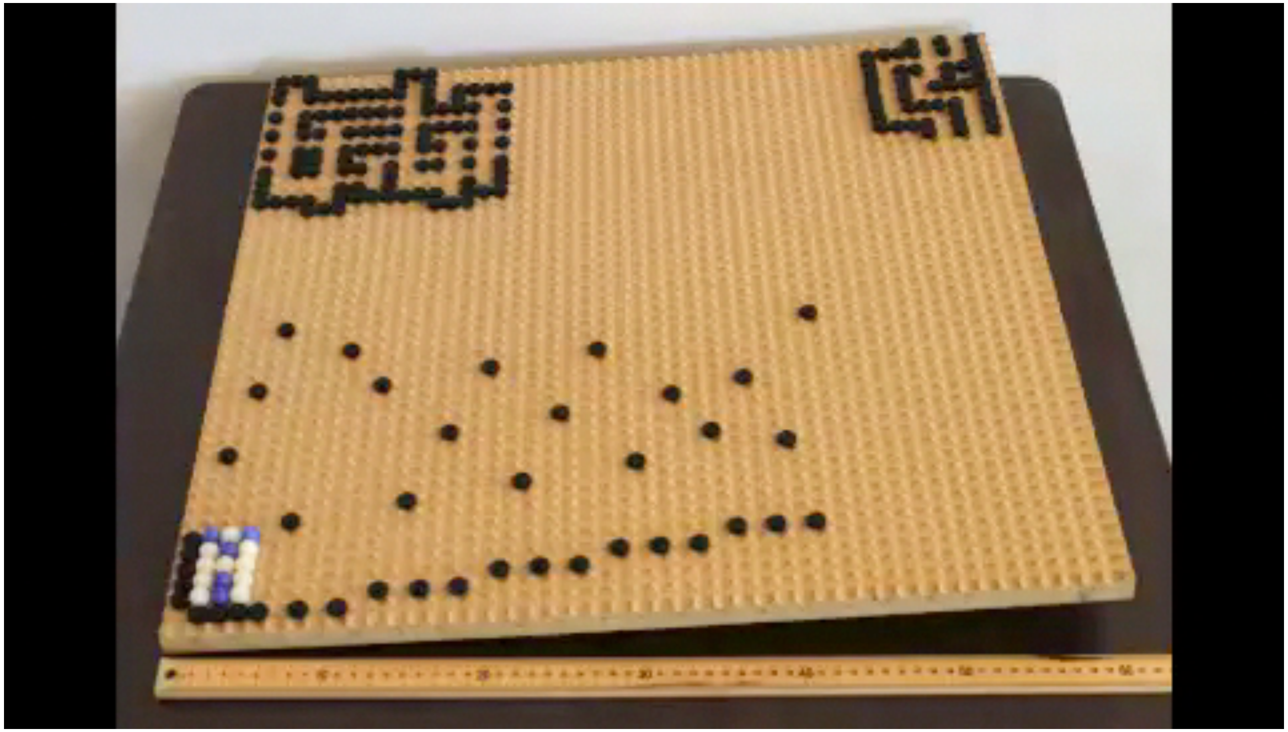
Demo with Real Objects



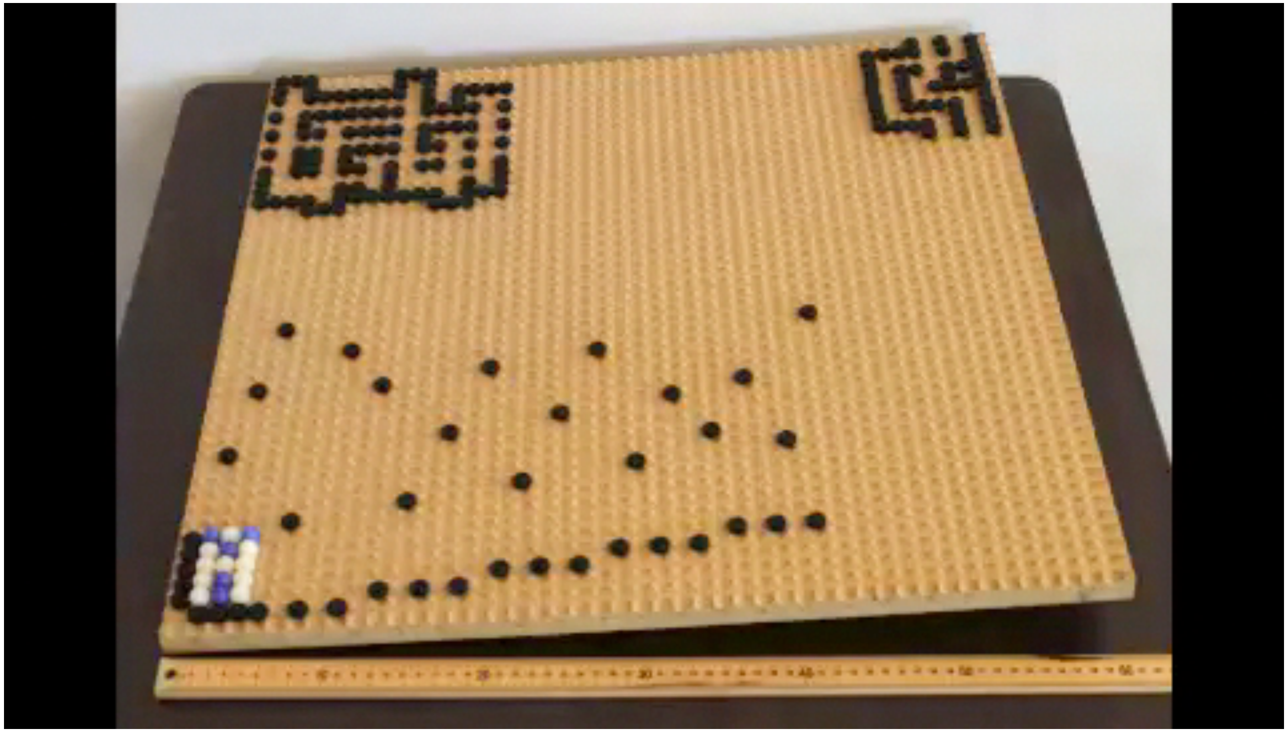
Demo with Real Objects



Demo II



Demo II



Conclusions

Conclusions

- More work in theory and practice!

Thank you!