MALMÖ UNIVERSITY

Online Strategies with Applications to Search and Exploration

Bengt J. Nilsson

Malmö University

Ljubljana Summer School 2014

Overview

- Online problems and competitive analysis
- Some hopefully explanatory examples
- History of search and exploration problems
- Searching for a target on a line, multiple agents on several lines
- Searching for a target in a monotone and a street polygon
- Exploration problems: overview
- Exploring rectilinear polygons
- Exploring simple polygons (problems)
- Exploring trees with multiple agents

Online Problems

What is an *Online Problem*?

> "A (computational) problem where the input is revealed piecemeal (in little chunks) as the computation develops. Need to make decisions without knowledge of future input."

I call such decision sequences *strategies* rather than algorithms.

How do you measure the performance of a strategy for an online problem with respect to an optimization criterion?

Competitive Analysis

$S(I)$ is the output of the strategy after input I

$OPT(I)$ is the optimum output after input I

$c(\cdot)$ is the optimization criterion (we assume minimization)

We say that strategy S is R-competitive, if

$$c(S(I)) \leq R \cdot c(OPT(I)) + Q,$$

for all input sequences I. (R is the competitive factor/ratio)

This is equivalent to

$$\frac{c(S(I))}{c(OPT(I))} \leq R + \epsilon$$

for all $\epsilon > 0$ with input sequences I having cost sufficiently large

Example: Ski Rental Problem

Problem: You are going skiing for the first time and need skis. You know that each time you go it will be for a full week but you don't know how many times you will go. You have two options:

1. Rent skis for €60 per week
2. Buy skis for €600, we assume then they last a lifetime

Optimization criterion is to minimize the cost for skis

Give a strategy with smallest competitive factor for the ski rental problem!

Example: Ski Rental Problem

Optimum cost:

Week	0	1	2	3	4	5	6	7	8	9	10	11	...
€	0	60	120	180	240	300	360	420	480	540	600	600	...

Analysis uses an *adversary*  to decide how many times you go skiing.

You will see him many times during this presentation

Example: Ski Rental Problem, cont'd

Here are two possible strategies:

If you buy skis before week 1,  decides that you never go skiing, competitive factor is $\frac{600}{0} = \infty$

If you decide to always rent skis,  decides you go skiing often, competitive factor is $\frac{n \cdot 60}{600} \to \infty$, as n increases

Example: Ski Rental Problem, cont'd

Here's a set of strategies:

You rent skis until week k and buy skis before week $k+1$, $k \geq 0$

 decides you go skiing k times is the worst case for this strategy, since strategy only pays $k \cdot 60 + 600$.  pays $k \cdot 60$, if $k < 10$ and 600 otherwise

k	0	2	4	6	8	9	10	11	12	14	16	18	...
R	∞	6	3.5	2.67	2.25	2.11	2.00	2.1	2.2	2.4	2.6	2.8	...

Best strategy is to rent skis 10 times, then buy skis

Example: Ski Rental Problem, cont'd

Generalize the ski rental problem:

1. Rent skis for €x per week
2. Buy skis for €y. Assume that $y > x > 0$.

Strategies:

You rent skis until week k and buy skis before week $k+1$, $k \geq 0$

Best strategy is to rent skis $k \leq \dfrac{y}{x}$ times, then buy skis when $k > \dfrac{y}{x}$

Guarantees competitive factor $\dfrac{\lfloor \frac{y}{x} \rfloor x + y}{\lfloor \frac{y}{x} \rfloor x} \to 2$

No strategy does better (we have looked at all of them!)

A Data Structure Problem: List Accessing

Sleator, Tarjan. 1985

Problem: You have a list structure with m elements and an online sequence σ of requests for objects in the list. Maintain the list organized so that the work of fulfilling the requests is minimized.

Operations allowed on the list

1. Accessing x in position j costs j. After an access you are allowed to move x to any point closer to the head of the list for free
2. Any other move costs i if an element is moved i positions

Optimization criterion is to minimize the total cost for the accesses

Example

Here's a short list

1	2	3	4	5	6
11	24	18	14	39	12

$\sigma = 24, 14, 39, 14, 14$

No restructuring: cost $2 + 4 + 5 + 4 + 4 = 19$

We look at some other strategies

TRANS(σ) (transpose) moves requested item one step closer to head

FC(σ) (frequency count) maintains list sorted on frequency

MTF(σ) (move to front) each retrieval moves element to head

OPT(σ) (optimal) not an online strategy

Example, cont'd

TRANS(σ)

	1	2	3	4	5	6	
24	11	24	18	14	39	12	2
14	24	11	18	14	39	12	4
39	24	11	14	18	39	12	5
14	24	11	14	39	18	12	3
14	24	14	11	39	18	12	2
	14	24	11	39	18	12	

cost $2 + 4 + 5 + 3 + 2 = 16$

What if the sequence is $\sigma = 12, 39, 12, 39, 12, 39, 12, 39, \ldots$?

Example, cont'd

FC(σ)

	1	2	3	4	5	6	
24	11_0	24_0	18_0	14_0	39_0	12_0	2
14	24_1	11_0	18_0	14_0	39_0	12_0	4
39	24_1	14_1	11_0	18_0	39_0	12_0	5
14	24_1	14_1	39_1	11_0	18_0	12_0	2
14	14_2	24_1	39_1	11_0	18_0	12_0	1
	14_3	24_1	39_1	11_0	18_0	12_0	

cost $2+4+5+2+1=14$

Drawback: need to maintain one extra counter per element

Example, cont'd

MTF(σ)

	1	2	3	4	5	6	
24	11	24	18	14	39	12	2
14	24	11	18	14	39	12	4
39	14	24	11	18	39	12	5
14	39	14	24	11	18	12	2
14	14	39	24	11	18	12	1
	14	39	24	11	18	12	

cost $2 + 4 + 5 + 2 + 1 = 14$

Example, cont'd

OPT(σ)

	1	2	3	4	5	6	
24	11	24	18	14	39	12	2
14	11	24	18	14	39	12	4
39	14	11	24	18	39	12	5
14	14	11	24	18	39	12	1
14	14	11	24	18	39	12	1
	14	11	24	18	39	12	

cost $2 + 4 + 5 + 1 + 1 = 13$

OPT(σ) not efficiently computable even if σ is known in advance

Analysis of MTF

Digress to define a new concept

Definition: Given lists L_1 and L_2, an *inversion* is a pair occuring in *different* order in the two list.

	1	2	3	4	5	6
L_1	11	24	18	14	39	12
L_2	14	39	24	11	18	12

$\Phi(L_1, L_2)$ denotes the total number of inversion between two lists

$\Phi(L_1, L_2) = 7$ in our example above

$\Phi(L, L) = 0$ always

Analysis of MTF, cont'd

t_i is the cost of request i in MTF(σ)

Φ_i number of inversions between MTF(σ) list and OPT(σ) list after i requests

$a_i = t_i + \Phi_i - \Phi_{i-1}$ amortized cost of request i in MTF(σ)

$c(\mathrm{opt}_i(\sigma))$ cost of request i in optimal strategy OPT(σ)

We have

$$c(\mathrm{MTF}(\sigma)) = \sum_{i=1}^{|\sigma|} t_i = \sum_{i=1}^{|\sigma|} a_i - \Phi_i + \Phi_{i-1} = \underbrace{\Phi_0}_{=0} - \underbrace{\Phi_{|\sigma|}}_{\geq 0} + \sum_{i=1}^{|\sigma|} a_i \leq \sum_{i=1}^{|\sigma|} a_i$$

Analysis of MTF, cont'd

Look at cost for request i accessing element x in position j in OPT

	1							k				m
MTF_{i-1}		*	*		*			x				
MTF_i	x		*	*		*						

	1					j						m
OPT_{i-1}						x	*		*	*		

Let v denote the number of items before x in MTF but after x in OPT

Thus, $k - v - 1 \leq j - 1 \Rightarrow k - v \leq j$

$a_i = t_i + \Delta\Phi_i = k - v + (k - v - 1) = 2(k - v) - 1 \leq 2j - 1 = 2 \cdot c(\mathrm{opt}_i(\sigma)) - 1$

Analysis of MTF, cont'd

Hence,

$$c(\mathrm{MTF}(\sigma)) \leq \sum_{i=1}^{|\sigma|} a_i \leq \sum_{i=1}^{|\sigma|} 2 \cdot c(\mathrm{opt}_i(\sigma)) - 1 = 2 \cdot c(\mathrm{OPT}(\sigma)) - |\sigma|$$

Since $c(\mathrm{OPT}(\sigma)) \leq |\sigma| m$ (m length of list)

$$c(\mathrm{MTF}(\sigma)) \leq (2 - \frac{1}{m}) c(\mathrm{OPT}(\sigma))$$

A Memory Allocation Problem: Paging

Sleator, Tarjan. 1985

Problem: You have k blocks (pages) of fast memory (RAM/cache) and m blocks of slow memory (disk). (Paging emulates large internal memory sizes.) σ is a sequence of memory requests, if ...

1. the request is in cache, nothing happens
2. the request is not in cache, place it in empty page if you can or $\rightarrow$ *page fault*.

   We need to evict a page to place the new one in its place

Optimization criterion is to minimize the number of page faults

Well-studied problem since the 60's!

Paging Example

Several paging schemes exist, LRU (Least-Recently-Used) probably best known

Example, assume cache (4 pages) has already been filled

Maintain list in LRU order (Emulates MTF list accessing)

Don't implement like this!

$\sigma = 12, 39, 14, 18, 24, 39, 11, 16, 39, 14$

	1	2	3	4	
24	18	14	39	12	1
39	24	18	14	39	
11	39	24	18	14	1
16	11	39	24	18	1
39	16	11	39	24	
14	39	16	11	24	1
	14	39	16	11	

cost 4

Paging Example, cont'd

Optimal paging scheme, LFD (Longest-Forward-Distance) not an online strategy

Example, assume cache (4 pages) has already been filled

$\sigma = 12, 39, 14, 18, 24, 39, 11, 16, 39, 14$

	1	2	3	4	
24	12	39	14	18	1
39	12	39	14	24	
11	12	39	14	24	1
16	11	39	14	24	1
39	16	39	14	24	
14	16	39	14	24	
	16	39	14	24	

optimal cost 3

Paging Lower Bound

k is size of cache. Assume $m = k + 1$ pages in total

For any request sequence σ, $c(\mathrm{LFD}(\sigma)) \leq \frac{|\sigma|}{k}$

Proof: Assume LFD(σ) evicts page p in step i. All other pages must be requested at least once before page p is requested again. Hence, next fault is earliest in step $i + k$ ∎

For any online strategy S, there is a request sequence σ so that $S(\sigma)$ faults on every request

Proof:  constructs σ by always choosing to request the page currently not in cache ∎

$\Rightarrow$ S is at best k-competitive

Paging Upper Bound

k is size of cache

LRU(σ) is k-competitive

Proof: Subdivide σ into *maximal sequences* of k different page requests, $\sigma = \sigma_1, \ldots, \sigma_L$.

Each σ_i must obey $|\sigma_i| \geq k$.

LRU incurs at most k faults in each σ_i since it contains only k different requests.

For any σ_i, let p be the first page requested in σ_i and q is the first page requested in σ_{i+1}, $q \notin \sigma_i$. Together, $\sigma_i \cup \{q\}$ contains $k+1$ different page requests so *any* strategy, including OPT, must incur at least one page fault in σ_i ■

Paging, cont'd

Experimental studies show LRU to be constant-competitive for any (sufficiently large) cache sizes

Competitive analysis *is not* suitable to accurately model paging behavior. Depend heavily on *locality*

Better analysis using randomization?

In randomized analysis,  does not see the strategies exact moves but only the probabilities for the different move. Defines different classes of 

Paging, the MARK Strategy

Here's a randomized strategy: Fiat, et al. 1988

MARK (Each page has a mark bit, initially 1)

Page request p:

If p is in cache, then $\text{mark}(p) := 1$

If p is not in cache, then evict random page q s.t. $\text{mark}(q) = 0$

If all pages are marked, then unmark them and try again

replace q by p and $\text{mark}(p) := 1$

Analysis of MARK Strategy

Subdivide σ into sequences between which MARK resets all page bits, $\sigma = \sigma_1, \ldots, \sigma_L$

For each σ_i, a page is *old*, if it is in the cache when σ_i starts. All other pages are *new*

Assume σ_i contains n_i new page requests $\Rightarrow \leq n_i$ page faults

A new page never gets evicted in σ_i

Assume jth request to old page requested the first time is in cache with probability $\geq \dfrac{k-j+1-n_i}{k-j+1}$ (worst case = when all n_i new page faults occur first) $\Rightarrow$ not in cache with probability $\leq \dfrac{n_i}{k-j+1}$

Analysis of MARK Strategy, cont'd

Expected number of page faults for MARK in σ_i is

$$\leq \underbrace{n_i}_{\text{new}} + \underbrace{\sum_{j=1}^{k-n_i} \frac{n_i}{k-j+1}}_{\text{old}} = n_i + n_i \sum_{j=n_i+1}^{k} \frac{1}{j}$$

$$= n_i + n_i(H_k - H_{n_i})$$

$$= n_i(H_k - H_{n_i} + 1) \leq n_i H_k$$

$H_k = \sum_{j=1}^{k} 1/j \leq 1 + \ln k$ is kth harmonic number

Total number of page faults is $\leq H_k \sum_{i=1}^{L} n_i$

Analysis of MARK Strategy, cont'd

Number of page faults for OPT in $\sigma_{i-1}\sigma_i$ (two consecutive sequences) is $\geq n_i$ ($\geq n_1$ in σ_1)

We have $\geq \sum_{i=1}^{L/2} n_{2i-1}$ and $\geq \sum_{i=1}^{L/2} n_{2i}$ page faults

Hence, the total number of page faults is

$$\geq \sum_{i=1}^{L/2} (n_{2i-1} + n_{2i})/2 = \sum_{i=1}^{L} n_i/2$$

Competitive factor becomes $2H_k \leq 2 + 2\ln k$ ■

No randomized paging strategy has competitive factor $< H_k$

History of Search and Exploration Problems

- Theseus and the Minotaur, antiquity
- 18th Century Garden Labyrinths (Versailles)
- Childrens Games, tag/search
- Mathematical Pursuit Games, (Bouger's Pirate Ship, 1732)
- Rufus Isaac, Differential Games, 1965
- Shmuel Gal, Search Games, 1980, (Alpern, Gal 2003)
- Numerous isolated articles...

Searching on a Line

Beck 1964; Bellman 1964; Gal 1980; Baeza-Yates *et al.* 1993

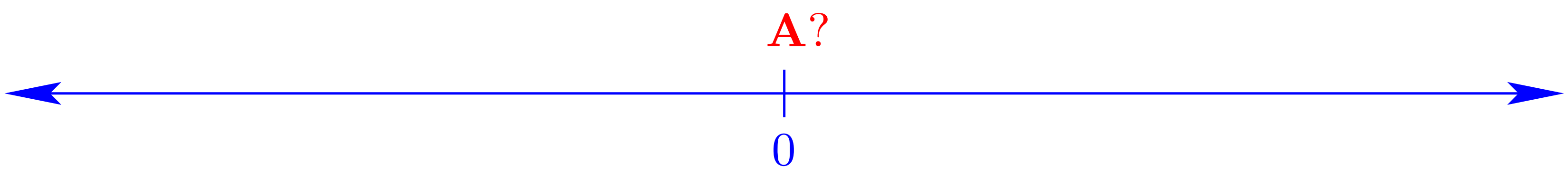
Problem: A cow knows that the farmer puts hay at some position along an (infinitely long) fence

What is the best strategy for the cow to find the hay if she wants to walk the minimum distance? The cow only sees the hay when she stands on it (It is foggy!)

Linear Search Problem A target (point) is placed somewhere along an infinite line. An agent (point) stands at the origin.

What is the best search strategy for the agent to find the target?

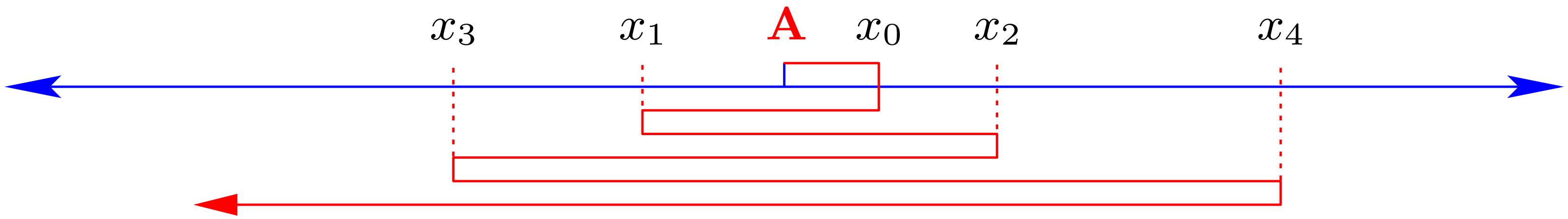
Searching on a Line, cont'd



Go left or go right? . . . and how far? Where is the target?

What specifies a strategy?

Searching on a Line, cont'd



Go left or go right? . . . and how far? Where is the target?

What specifies a strategy?

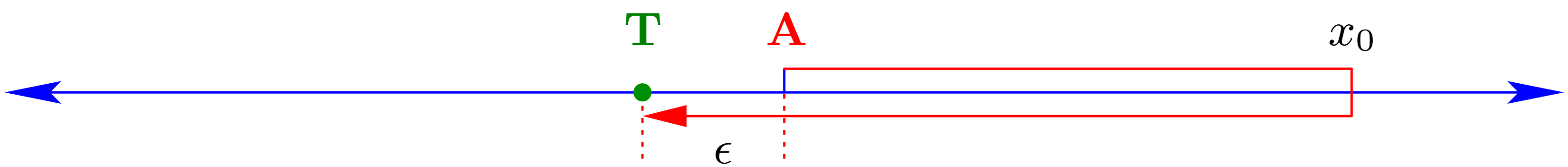
A sequence of turning points: $x_0, x_1, \ldots$

Even indices go to the right of 0, odd to the left

$x_i \leq x_{i+2}$, otherwise makes no sense

NOTE: We compare length of strategy to distance D to target

Searching on a Line, cont'd



Immediate Problem: What if $D = \epsilon \ll x_0$?

No strategy can be competitive then!!!

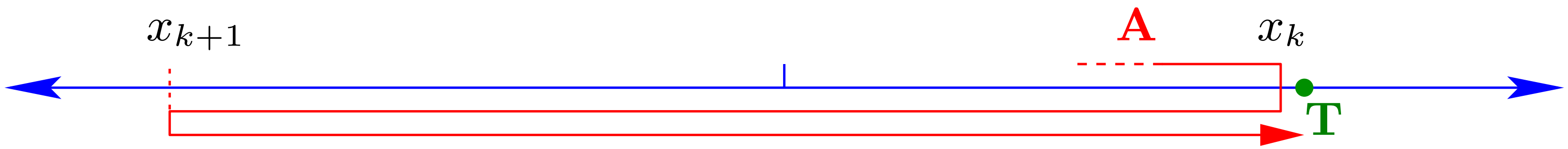
Three solutions:

1. Use extra additive term in competitive formula,
   $c(S) \leq R \cdot c(OPT) + Q$
2. Require that D is bounded from below by known amount, $D > 1$
3. Allow strategy to *wiggle* in the beginning, cost is $\sum_{i=-\infty}^{?} 2x_i$

Searching on a Line, cont'd

Let R_k be the worst case competitive factor given that $D \in]x_k, x_{k+2}]$, for each k.

$$R_k(X_{k+1}) \stackrel{\text{def}}{=} \sup_{D \in]x_k, x_{k+2}]} \frac{D + \sum_{i=0}^{k+1} 2x_i}{D} = 1 + 2\frac{\sum_{i=0}^{k+1} x_i}{x_k}$$



 places target after the turn at x_k so that **A** just misses it

$R_k(X_{k+1})$, ($X_{k+1} = x_0, \ldots, x_{k+1}$, each $x_i > 0$), is a *functional*, a mapping from a vector space to a scalar field

Searching on a Line, cont'd

Theorem: Gal 1980. If, for a set of functionals $F_k(X_k)$ ($X_k = x_0, \ldots, x_k$, each $x_i > 0$), each satisfies

1. $F_k(X_k)$ is continuous
2. $F_k(\alpha X_k) = F_k(X_k)$, for all $\alpha > 0$, absorbing (homogeneous)
3. $F_k(X_k + Y_k) \le \max\{F_k(X_k), F_k(Y_k)\}$ for all X_k, Y_k, unimodal
4. $\liminf_{t\to\infty} F_k(1/t^k, \ldots, 1/t, 1) = \liminf_{\epsilon_k,\ldots,\epsilon_1\to 0} F_k(\epsilon_k, \ldots, \epsilon_1, 1)$
5. $\liminf_{t\to 0} F_k(1, t, \ldots, t^k) = \liminf_{\epsilon_1,\ldots,\epsilon_k\to 0} F_k(1, \epsilon_1, \ldots, \epsilon_k)$

and for any positive sequence $X = \{x_i\}_{i=0}^{\infty}$; $F_{k+1}(x_0, \ldots, x_{k+1}) \ge F_k(x_0, \ldots, x_k)$ there is a constant $a \ge 0$;

$$\sup_{k\ge 0} F_k(x_0, \ldots, x_k) \ge \sup_{k\ge 0} F_k(1, a, \ldots, a^k)$$

Searching on a Line, cont'd

The competitive factor is

$$\sup_{k\geq 0} R_k(X_{k+1}) = \sup_{k\geq 0} 1 + 2\frac{\sum_{i=0}^{k+1} x_i}{x_k} \overset{\textbf{Theorem}}{\geq} \sup_{k\geq 0} 1 + 2\frac{\sum_{i=0}^{k+1} a^i}{a^k}$$

$$= \sup_{k\geq 0} 1 + 2\frac{a^{k+2}-1}{a^{k+1}-a^k} = 1 + 2\frac{a^2}{a-1}$$

$$\geq \mathbf{9}$$

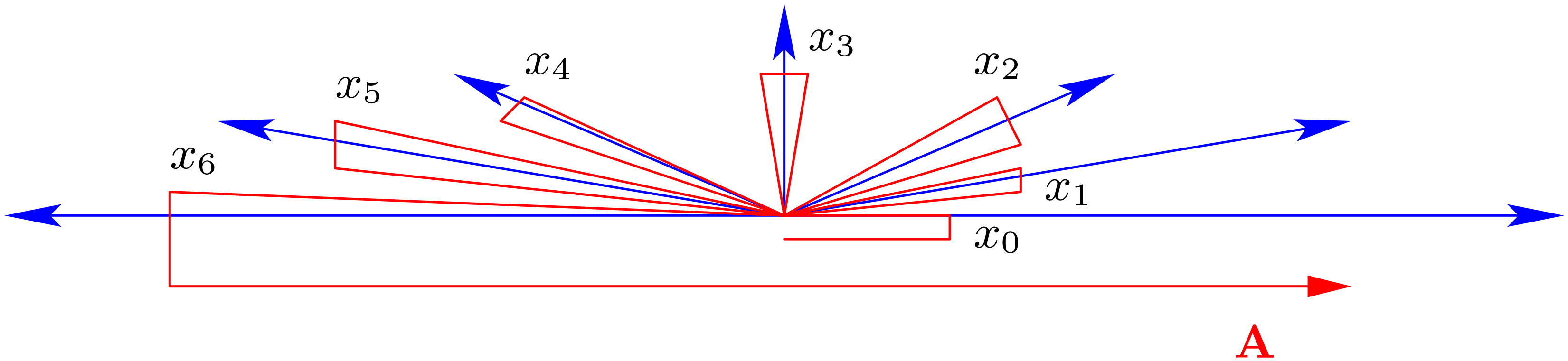
for any $a > 1$. Value is attained when $a = 2$ ■

Searching on a Line, cont'd

Strategy is called the *doubling strategy*. The smallest competitive ratio 9 is achieved when (abusing notation) $x_k = (-2)^k$ on the line, i.e., $1, -2, 4, -8, \ldots$ and this is optimal as we have seen.

Searching on m Rays

What if agent wants to search on m rays, all starting at the origin? ($m = 2$ we have done)



Visit rays in some cyclic order!

Searching on m Rays, cont'd

Let R_k^m be the worst case competitive factor given that $D \in]x_k, x_{k+m}]$, for each k.

$$R_k^m(X_{k+1}) \stackrel{\text{def}}{=} \sup_{D \in]x_k, x_{k+m}]} \frac{D + \sum_{i=0}^{k+m-1} 2x_i}{D} = 1 + 2\frac{\sum_{i=0}^{k+m-1} x_i}{x_k}$$

 places target after the turn at x_k so that **A** just misses it

Searching on m Rays, cont'd

The competitive factor is

$$\sup_{k \geq 0} R_k^m(X_{k+1}) = \sup_{k \geq 0} 1 + 2\frac{\sum_{i=0}^{k+m-1} x_i}{x_k} \overset{\textbf{Theorem}}{\geq} \sup_{k \geq 0} 1 + 2\frac{\sum_{i=0}^{k+m-1} a^i}{a^k}$$

$$= \sup_{k \geq 0} 1 + 2\frac{a^{k+m} - 1}{a^{k+1} - a^k} = 1 + 2\frac{a^m}{a-1}$$

$$\geq 1 + 2\frac{m^m}{(m-1)^{m-1}} \approx 1 + 2e \cdot m$$

for any $a > 1$. Value is attained when $a = \dfrac{m}{m-1}$ ■

Generalizes the doubling strategy, *Exponential Search* also optimal since visiting order doesn't matter

Parallel Searching on m Rays

Hammar, Nilsson, Schuierer 2001

Problem: Parallel Search Problem A target (point) is placed somewhere on m rays,, all starting at the origin. m agents (points) are available.

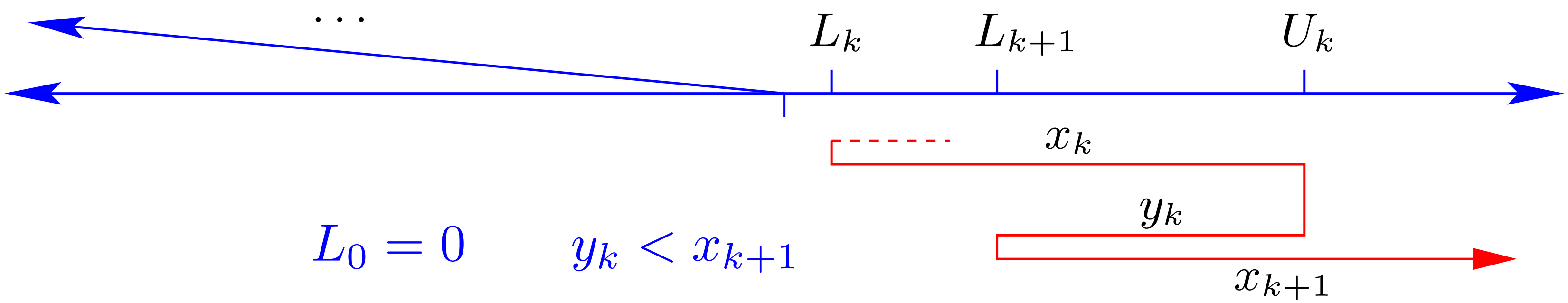
Agents can communicate only when they meet

What is the best search strategy for all the agents to reach the target?

We look at two cases: *symmetric strategies* and *monotone strategies*

Parallel Searching on m Rays: Symmetric Case

Each agent performs the same moves on its ray until target is found



$$L_0 = 0 \qquad y_k < x_{k+1}$$

$$L_k = L_{k-1} + x_{k-1} - y_{k-1} = \sum_{i=0}^{k-1} x_i - y_i$$

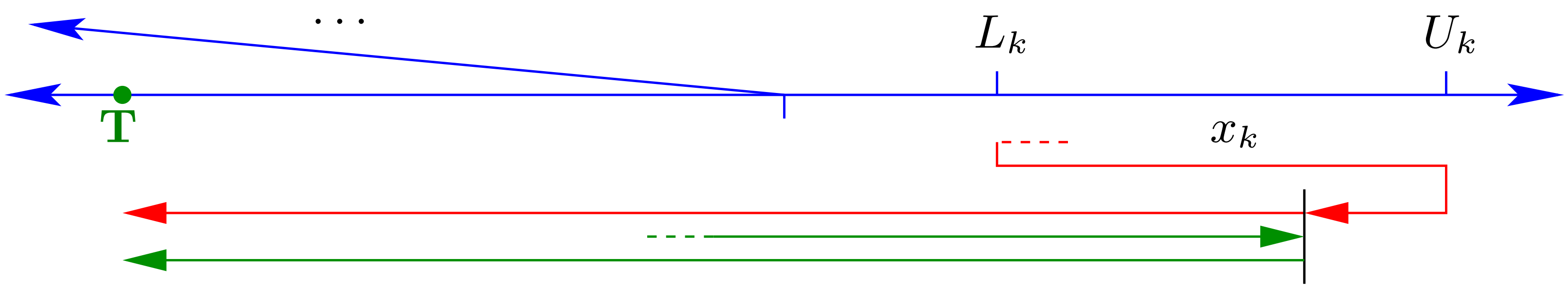
$$U_k = L_k + x_k = x_k + \sum_{i=0}^{k-1} x_i - y_i$$

$$d_k = \sum_{i=0}^{k} x_i + y_i \quad \text{(Total distance moved)}$$

Parallel Searching. . . : Symmetric Case, cont'd

We are in Step k when moving between L_k and L_{k+1}.

Look at the last agent informed of placement of target. Assume informed in Step k. Target found in Step $k-j$



$$R^S_{jk} = \sup_{D\in]U_{k-j-1},U_{k-j}]} \frac{\overbrace{d_{k-1}}^{\sum_{i=0}^{k-1} x_i+y_i} + x_k + \overbrace{U_k}^{x_k+\sum_{i=0}^{k-1} x_i-y_i} + D}{D}$$

Parallel Searching...: Symmetric Case, cont'd

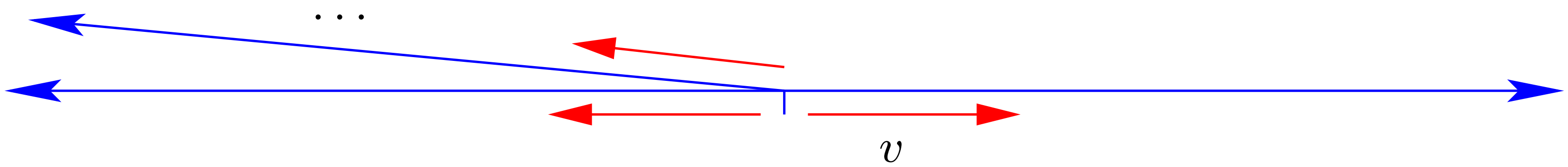
$$\begin{aligned}
\sup_{k\geq 0} R^S_{jk} &= \sup_{D\in]U_{k-j-1},U_{k-j}]} \frac{\sum_{i=0}^{k-1} x_i + y_i + x_k + x_k + \sum_{i=0}^{k-1} x_i - y_i + D}{D} \\
&= \sup_{D\in]U_{k-j-1},U_{k-j}]} 1 + 2\frac{\sum_{i=0}^{k} x_i}{D} \geq \sup_{k\geq 0} 1 + 2\frac{\sum_{i=0}^{k} x_i}{U_{k-j-1}} \\
&\geq \sup_{k\geq 0} 1 + 2\frac{\sum_{i=0}^{k} x_i}{x_{k-j-1}} \overset{\textbf{Theorem}}{\geq} \sup_{k\geq 0} 1 + 2\frac{\sum_{i=0}^{k} a^i}{a^{k-j-1}} = 1 + 2\frac{a^{j+2}}{a-1} \\
&\geq \mathbf{9}
\end{aligned}$$

for $j = 0$, attained for $a = 2$ $\blacksquare$

Parallel Searching on m Rays: Monotone Case

Each agent moves with speed $v \leq 1$ forward until target is found

We measure the *time* it takes for *all* agents to reach the target



Then informed agents (*hunters*) chase uninformed ones (*prey*) at full speed (=1) to communicate target

Step k starts when $\geq 2^k$ agents are hunters, ends when $\geq 2^{k+1}$ become hunters. Step k takes t_k time and t_F is the time for the last hunter to reach the target

$$R^M = \frac{D/v + \sum_{k=0}^{\log m - 1} t_k + t_F}{D}$$

Parallel Searching...: Monotone Case, cont'd

Step times form a geometric sequence

$$t_k \le t_{k-1} \left(\frac{1+v}{1-v}\right) = t_0 \left(\frac{1+v}{1-v}\right)^k = \frac{2D}{1-v}\left(\frac{1+v}{1-v}\right)^k$$

$$\sum_{k=0}^{\log m-1} t_k \le \frac{2D}{1-v} \sum_{k=0}^{\log m-1} \left(\frac{1+v}{1-v}\right)^k = \frac{D}{v}\left(\left(\frac{1+v}{1-v}\right)^{\log m} - 1\right)$$

$$t_F \le D\left(\frac{1+v}{1-v}\right)^{\log m} + D$$

Giving the competitive factor (only dependent on v)

$$R^M = \frac{1}{v} + \frac{1}{v}\left(\left(\frac{1+v}{1-v}\right)^{\log m} - 1\right) + \left(\frac{1+v}{1-v}\right)^{\log m} + 1 = 1 + \left(1 + \frac{1}{v}\right)\left(\frac{1+v}{1-v}\right)^{\log m}$$

$$= 1 + 2\frac{(\log m + 1)^{\log m + 1}}{(\log m)^{\log m}} \approx 1 + 2e(\log m + 1)$$

if we set $v = \frac{1}{1+2\log m}$ ■

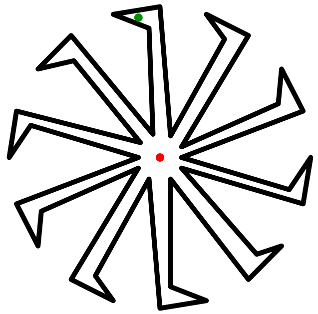
Searching for a Target in a Polygon

Klein 1992; Kleinberg 1994;
Icking, Klein, Langetepe, Schuierer, Semrau 2004

Problem: A point agent **A** is positioned at s in an unknown polygon. A point target t is placed somewhere in the polygon. **A** recognizes t when he *sees* it. The edges of the polygon act as obstacles and cannot be seen through. **A** has vision system, compass, memory

What is the best search strategy for the agent to find the target?

We compare length of **A**'s path to length of *shortest path* from s to t in polygon



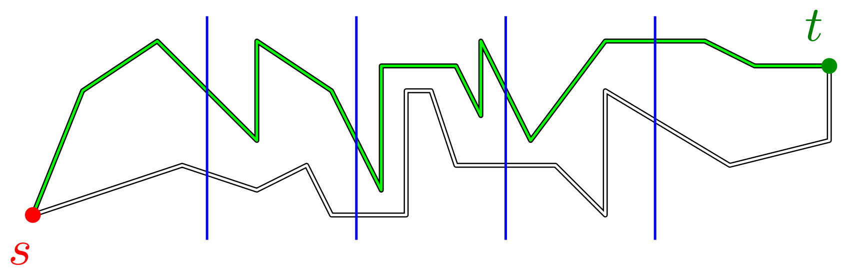
No constant competitive strategy exists in general


forces **A** to explore all tentacles ■

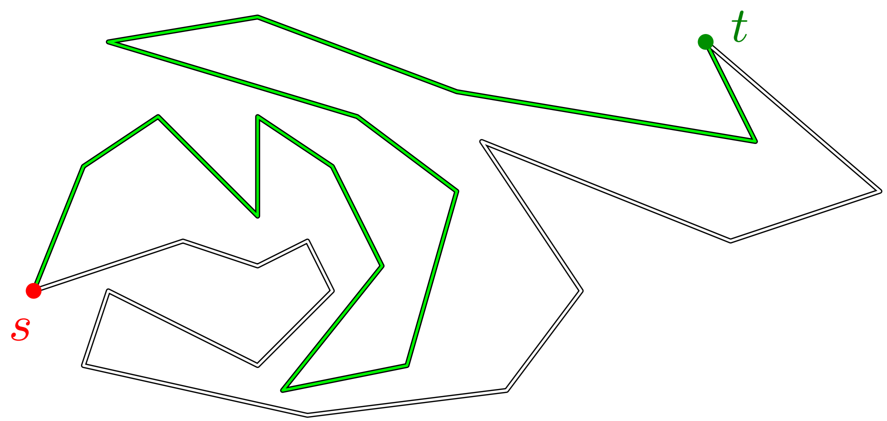
Searching for a Target in a Polygon

We look at special classes of polygons:

Monotone Polygons: boundary can be partitioned into two (x-)monotone chains. (Any intersection with vertical line is connected)



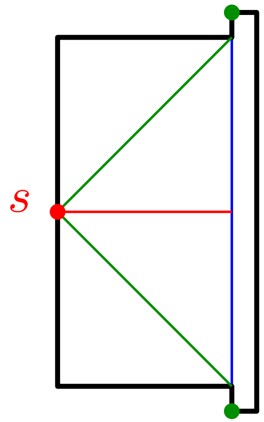
Street Polygons: boundary can be partitioned into two chains that *see* each other



Searching for a Target in a Polygon

A polygon is monotone $\Rightarrow$ it is a street

Lower Bound for both classes



Competitive factor is at least

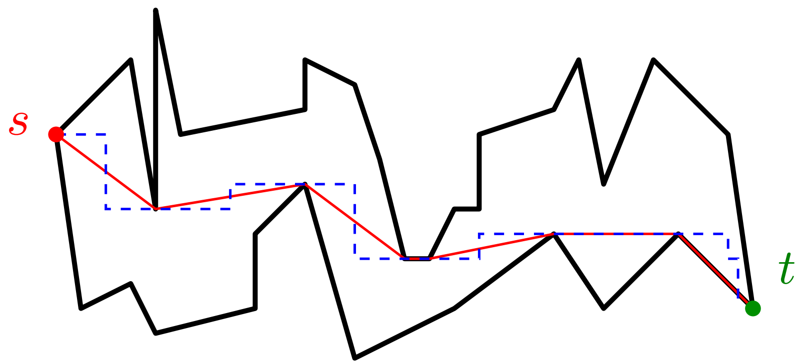
$$\frac{2D}{\sqrt{2}D} = \sqrt{2} = 1.414$$

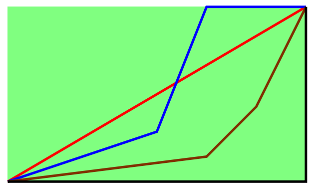
Searching for a Target in a Polygon

A Useful Trick!

Exchange shortest path from s to t by *shortest rectilinear path*

Path with edges parallel to coordinate axis





Length of *any* x- and y-monotone path is bounded by $\sqrt{2} \cdot ||$shortest path$||$

Path length increases by $\leq \sqrt{2}$ (Pythagorean theorem) ■

Change of metric from L_2 to L_1-metric

Searching for a Target in a Polygon

Upper Bound for Monotone Polygons

Strategy **MONOTONE SEARCH**

Move horizontally until:

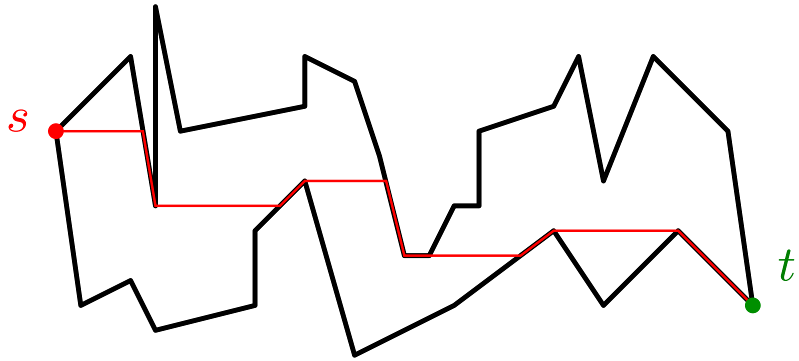
1) Agent sees the target; move to it; stop

2) Agent is on the boundary

if agent is on upper boundary, follow boundary down to vertex

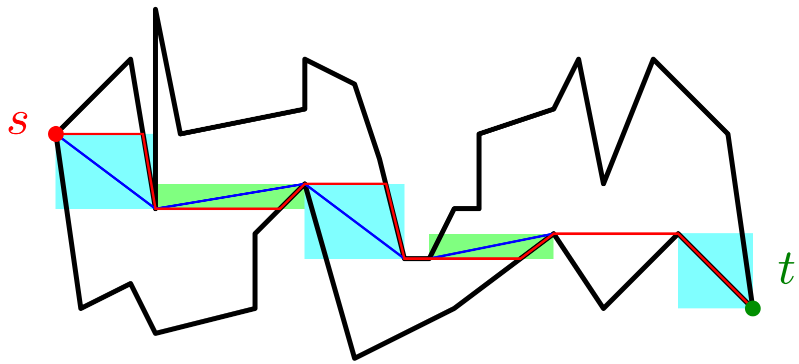
if agent is on lower boundary, follow boundary up to vertex

Repeat



Searching for a Target in a Polygon

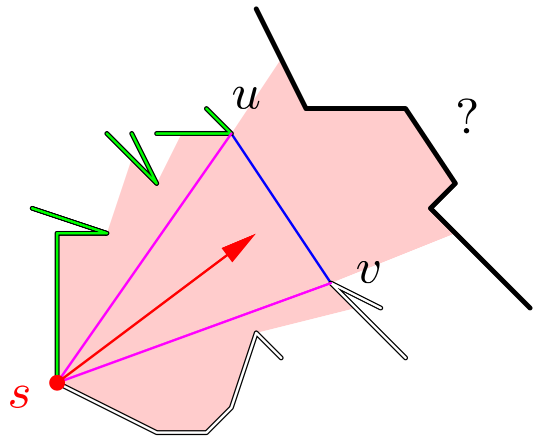
Upper Bound for Monotone Polygons



Strategy's path is contained in rectangles, also containing shortest path and each path is x- and y-monotone inside rectangles $\Rightarrow$ following rectangle boundary gives $\leq \sqrt{2} \Rightarrow$ competitive factor $\leq \sqrt{2}$ ■

Searching for a Target in a Polygon

Upper Bound for Street Polygons Kleinberg 1994



Left and right *windows* of *visibility polygon* must bound each of the two chains (green or white). Opposite boundary color is unknown. Maintain shortest path from s to two vertices u and v. Optimal path to target must pass one of them

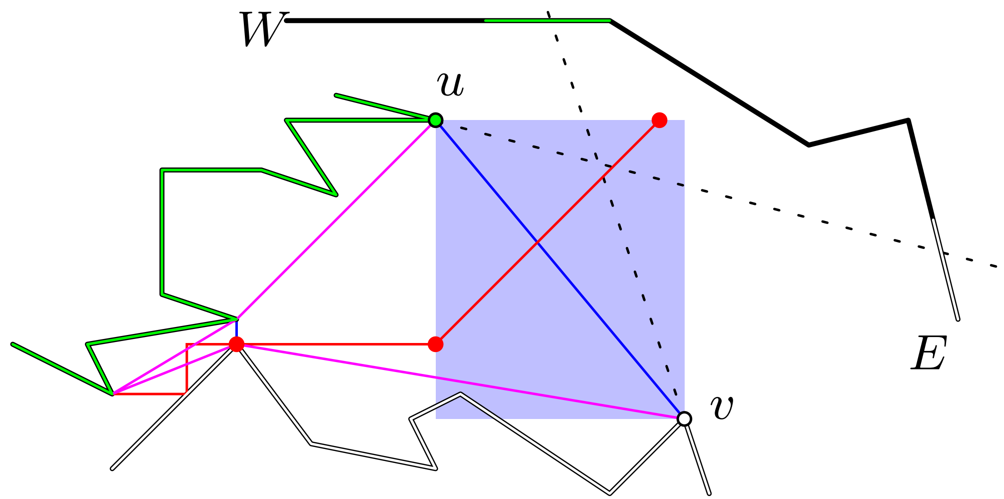
How do we move so that the detour is not too large?

Move rectilinearly! (or almost)

Searching for a Target in a Polygon

Upper Bound for Street Polygons

General view



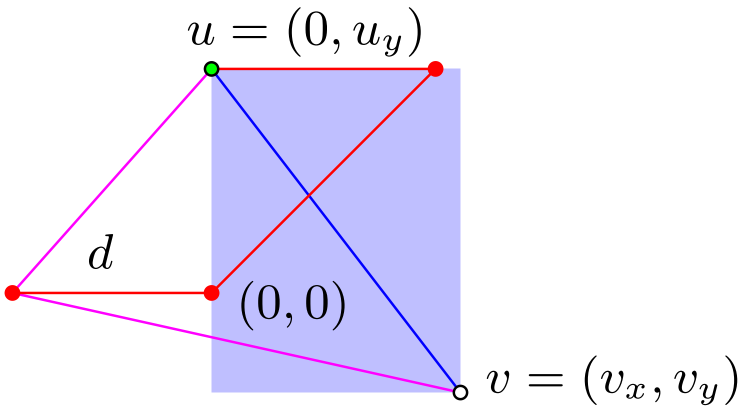
We reach bounding box spanned by u and v, go to 45° motion until…

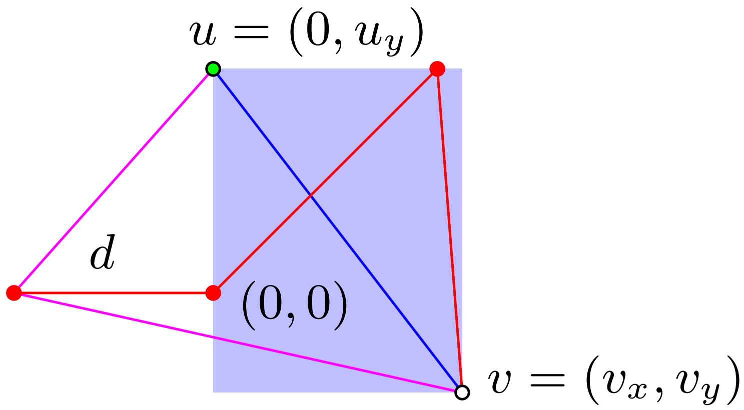
1. one of boundary parts W or E is completely seen
2. we exit box

We must now know which of u and v the optimal path to target passes.
Go to the appropriate vertex. Estimate the detour

Searching for a Target in a Polygon

Upper Bound for Street Polygons





Three cases:

1. we see target; go there!
2. shortest path goes through u, detour is $\dfrac{d + \sqrt{2}u_y + u_y}{\sqrt{d^2 + u_y^2}}$
3. it goes through v, detour is $\dfrac{d + \sqrt{2}u_y + \sqrt{(v_x - u_y)^2 + (v_y - u_y)^2}}{\sqrt{(d + v_x)^2 + v_y^2}}$

Searching for a Target in a Polygon

Upper Bound for Street Polygons

2. simpler case: $\dfrac{d+\sqrt{2}u_y+u_y}{\sqrt{d^2+u_y^2}}$, maximized for $u_y = d(1+\sqrt{2})$ (differentiate)

$$R_u \leq \frac{d+d(\sqrt{2}+1)^2}{\sqrt{d^2+d^2(1+\sqrt{2})^2}} = \sqrt{1+(1+\sqrt{2})^2} = \sqrt{4+\sqrt{8}} \approx 2.613$$

3. complicated case: $\dfrac{d+\sqrt{2}u_y+\sqrt{(v_x-u_y)^2+(v_y-u_y)^2}}{\sqrt{(d+v_x)^2+v_y^2}}$, worst case occurs when $v_x = u_y$ and $d = 0$, maximized for $v_y = v_x(1-\sqrt{2})$ (differentiate)

$$R_v \leq \frac{\sqrt{2}v_x+\sqrt{(v_x(1-\sqrt{2})-v_x)^2}}{\sqrt{v_x^2+v_x^2(1-\sqrt{2})^2}} = \frac{2\sqrt{2}}{\sqrt{4-2\sqrt{2}}} = \sqrt{4+\sqrt{8}} \approx 2.613$$

Searching for a Target in a Polygon

Upper Bound for Street Polygons

Fortunately ☺, both cases have the same relative detour. Since every subpath between consecutive pairs of vertices on the optimal path to target has detour, either R_u or R_v, the competitive ratio for Kleinberg's strategy is bounded by

$$R = \max\{R_u, R_v\} \leq \sqrt{4 + \sqrt{8}} \approx 2.613 \qquad \blacksquare$$

Icking, Klein, Langetepe, Schuierer, Semrau 2004 prove a strategy with optimal $\sqrt{2} \approx 1.414$ competitive factor

Exploration Problems

- Competitive search ⇒ exploration.  places target so the whole environment must be explored
- We will look at single agent exploration of:

  Polygons with holes (obstacles)

  Rectilinear simple polygons

  Simple polygons (short overview of problem)
- If there is time, we will consider multiple agent exploration in trees
- First, problem definition

The Exploration Problem

Problem: A point agent **A** is placed at s in an unknown environment and is required to explore it, i.e., make certain **A** sees all of it, can draw a map of it, and return to s. **A** has vision system, compass, memory

What is the best exploration strategy for the agent?

We compare length of **A**'s tour to length of *shortest tour* from s in the environment

Requirement of a tour is not (too much of) a restriction. Any path can be made into a tour by following shortest path back to s from other endpoint of path. Hence,

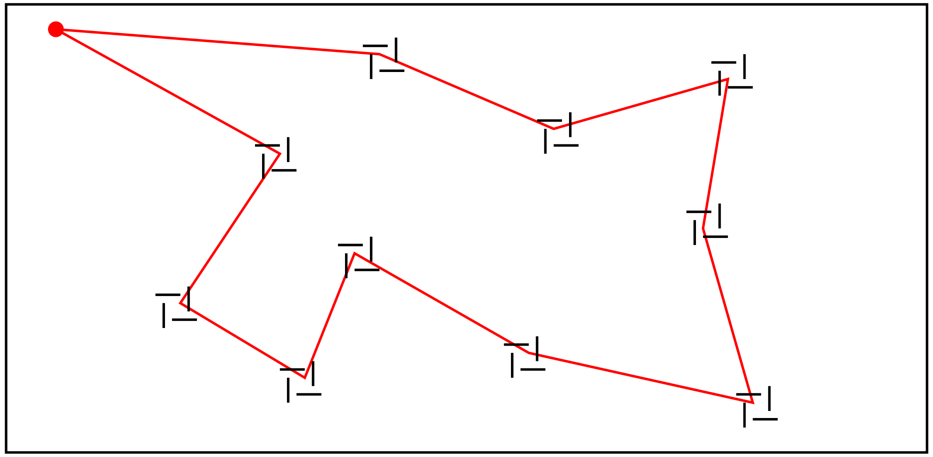
$$||\text{path}|| \leq ||\text{tour}|| \leq 2||\text{path}|| \qquad \text{(triangle inequality)}$$

What is the complexity situation for these problems?

The Offline Exploration Problem

Polygons with obstacles

We reduce from Geometric TSP proving NP-hardness (Chin, Ntafos 1988)



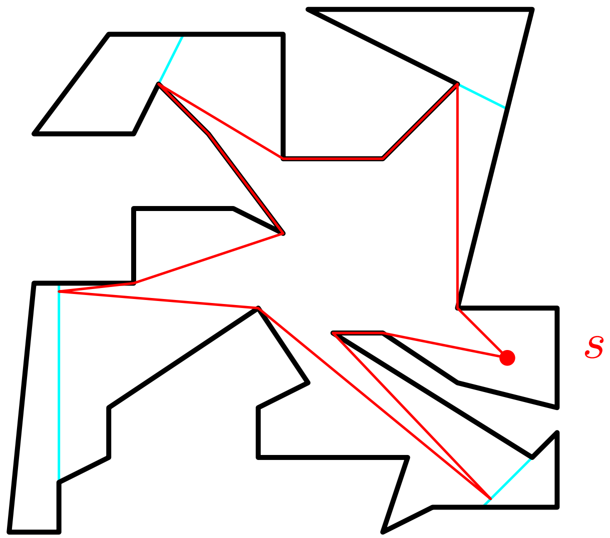
Mitchell 2013 proves $\Omega(\log n)$ inapproximable by reduction from Set Cover. n is number of vertices of polygon. Also shows $O(\log^2 n)$ approximation algorithm

Is there fixed parameter tractable algorithm for polygons with h holes? Complexity $O(f(h) \cdot n^c)$

The Offline Exploration Problem

Simple polygons The *shortest watchman tour* problem

Chin, Ntafos 1991; Tan, Hirata, Inagaki 1998; Dror, Efrat, Lubiw, Mitchell 2003

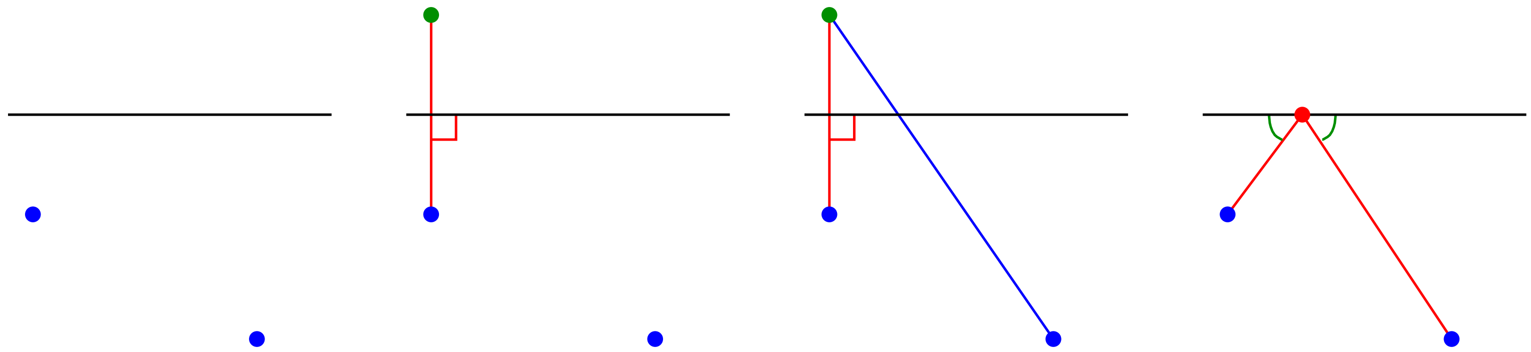


Transform problem to a shortest path problem.

The Shortest Watchman Tour Problem

Solution uses Heron's reflection principle.

Question: What is the shortest path between two points that visits a line?



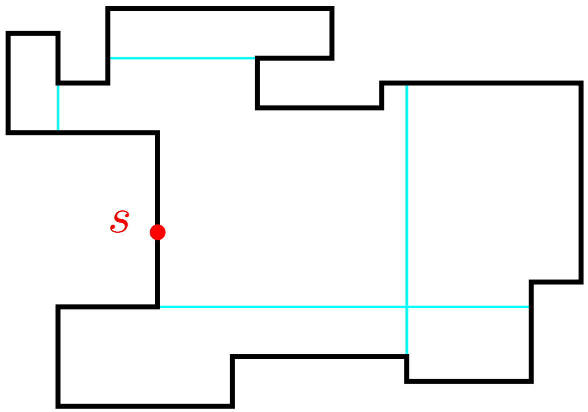
Reflect one point, compute the segment, fold the segment

Intersection on line at *perfect reflection point*

The Shortest Watchman Tour Problem, cont'd

We deal with rectilinear polygons having n vertices (Chin, Ntafos 1988)

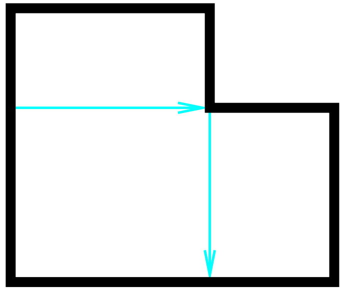
Assume for now that s is on the boundary!

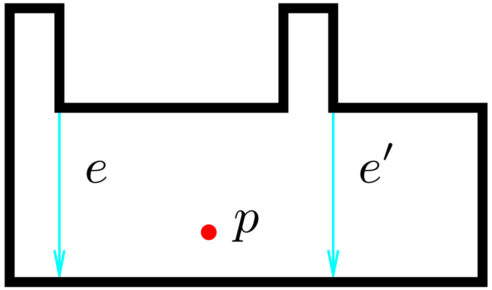


Claim 1: There are segments (extensions) such that if a tour visits them, the tour sees everything. (see next slide)

Claim 2: A shortest tour visits the extensions in the order they appear along the polygon boundary.

The Shortest Watchman Tour Problem, cont'd





e is a left extension
with respect to p

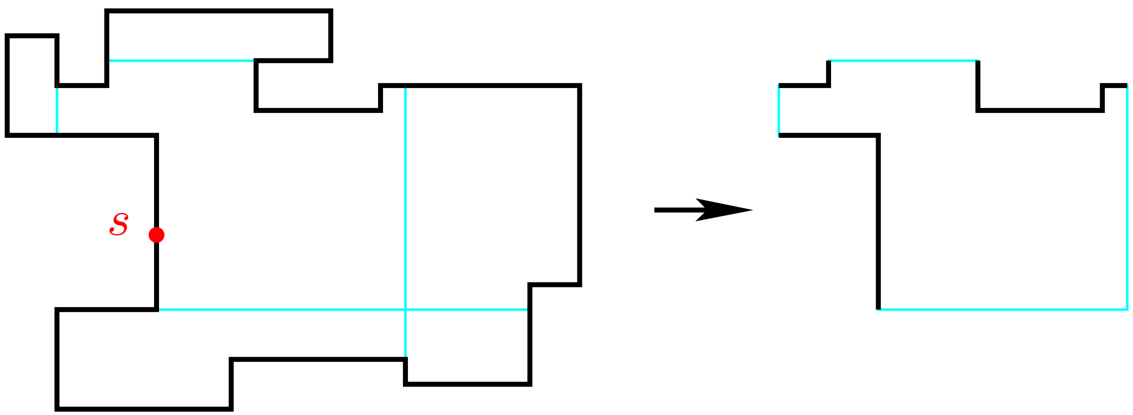
The *extensions* are the extended line segments of the sides of the polygon.

With respect to a point p, an extension is a *left extension* if p lies to the left of it.

A closed curve sees all of a simple polygon if and only if it has at least one point to the right of every extension.

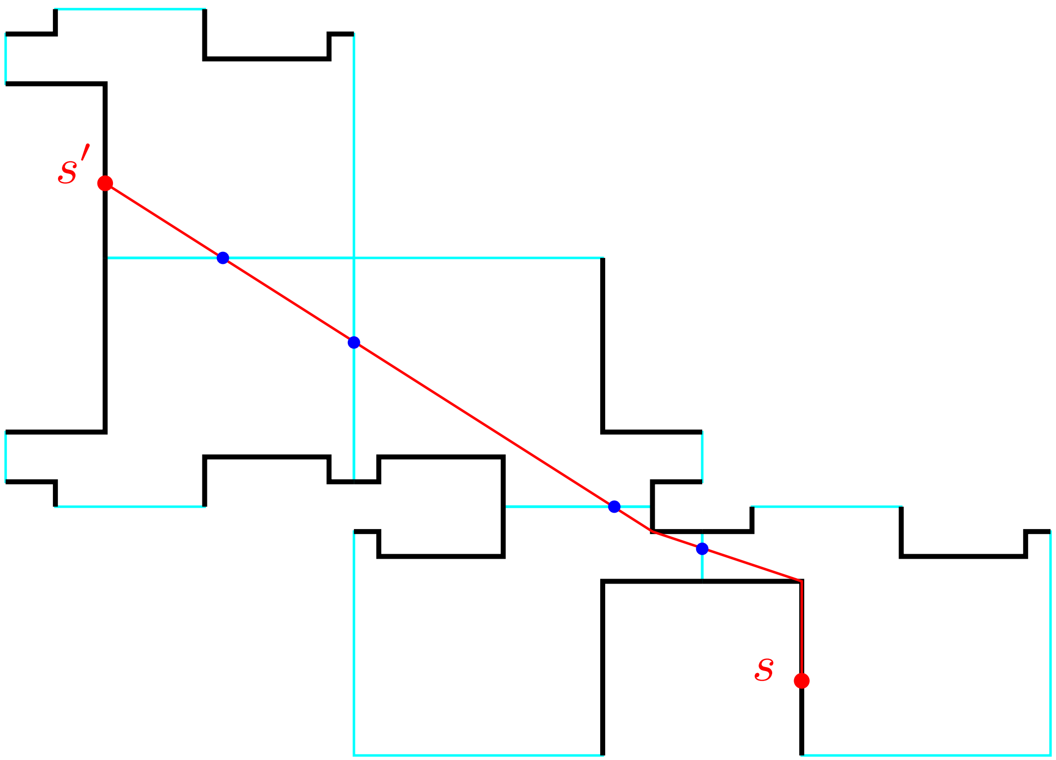
The Shortest Watchman Tour Problem, cont'd

Cut off unnecessary parts of the polygon.



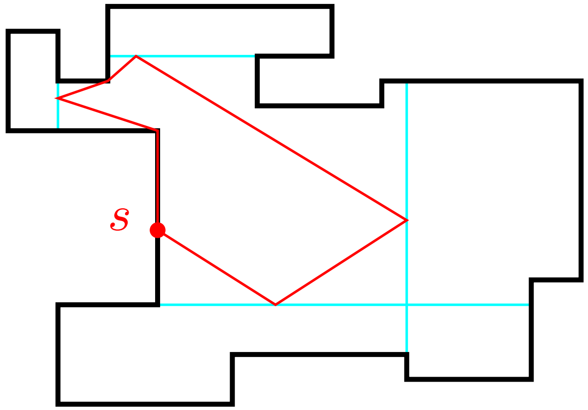
The Shortest Watchman Tour Problem, cont'd

Reflect the polygon using the extensions as mirrors, compute the shortest path from s to s' and establish the points of reflection



The Shortest Watchman Tour Problem, cont'd

Fold the path back to obtain a tour



The tour can be computed for a rectilinear simple polygon in $O(n)$ time ■

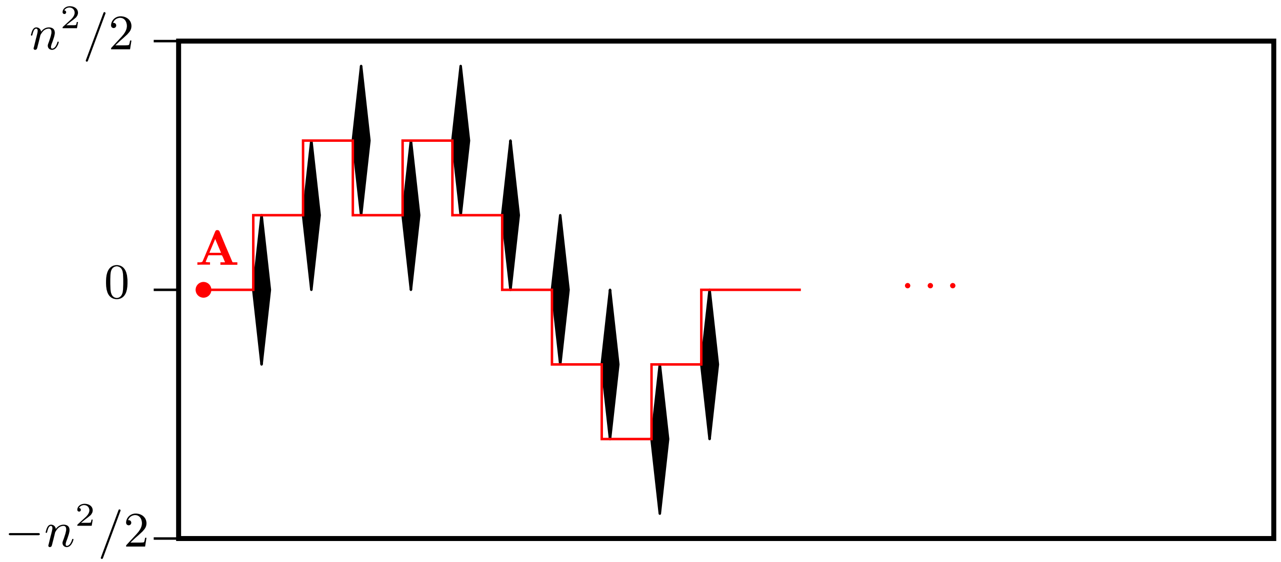
Similar ideas can be used to solve the shortest watchman problem for simple polygons

Problem: Not clear which extensions to reflect on and which to cross. Dynamic programming is used to choose the correct case. Best algorithm takes $O(n^3 \log n)$ time (Dror, Efrat, Lubiw, Mitchell 2003)

The Exploration Problem

Back to online exploration **Polygons with holes**

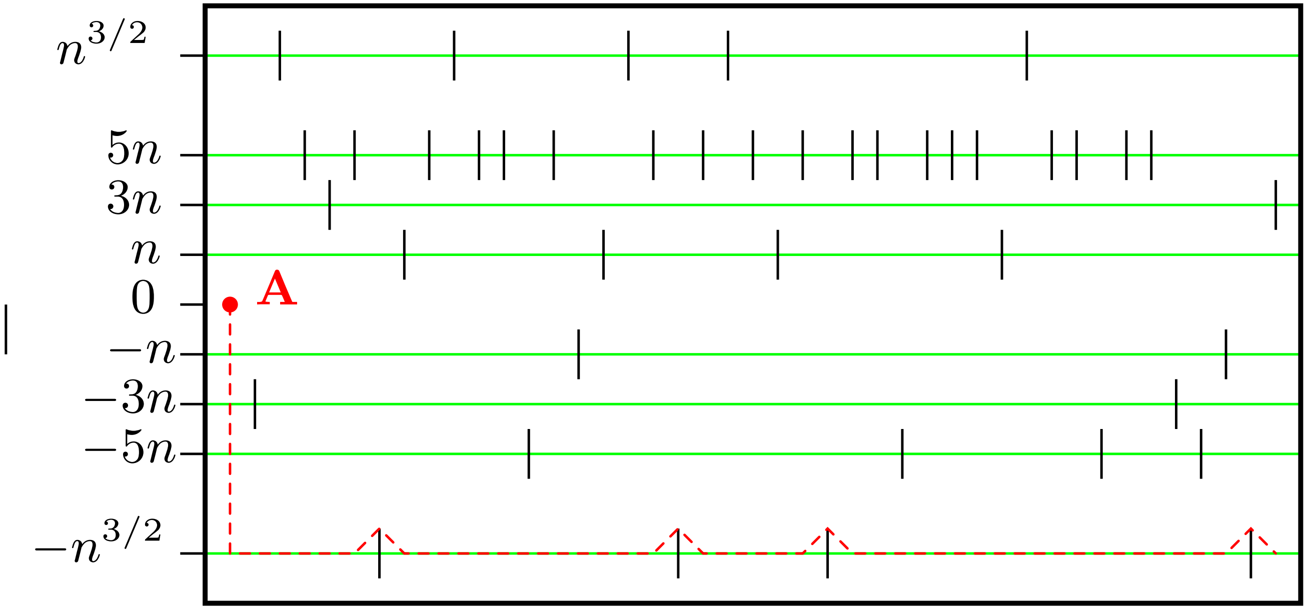
Remove tour requirement for awhile! Papadimitriou, Yannakakis 1991



In a rectangle of width $n+2$ and height n^2,  places n thin $\diamond$-shapes of height n in front of path, whichever direction **A** chooses to go. $\Rightarrow$ **A** moves at least distance $\frac{n}{2} \cdot n = n^2/2$

The Exploration Problem

Count how many $\diamond$ intersect each vertical line at height $n, -n, 3n, -3n, 5n, -5n, \ldots, \sqrt{n}n, -\sqrt{n}n$. A $\diamond$ can intersect at most one line. We have $\sqrt{n}$ lines. Not all such lines can intersect $> \sqrt{n}$ $\diamond$s otherwise we have $> \sqrt{n} \cdot \sqrt{n} = n$ obstacles. (Pigeonhole principle)



Exists a path of length $\leq 2n^{3/2}$. Competitive factor $\geq \sqrt{n}/4 = \Omega(\sqrt{n})$

Exploring Rectilinear Polygons

Deng, Kameda, Papadimitriou 1991; Kleinberg 1994; Hammar *et al.* 2003

Problem: A point agent **A** is placed at s in a *rectilinear simple polygons* and is required to explore it, and return to s. **A** has vision system, compass, memory

Distance is measured in L_1-metric, allows us to impose rectilinear motion

What is the best exploration strategy for the agent?

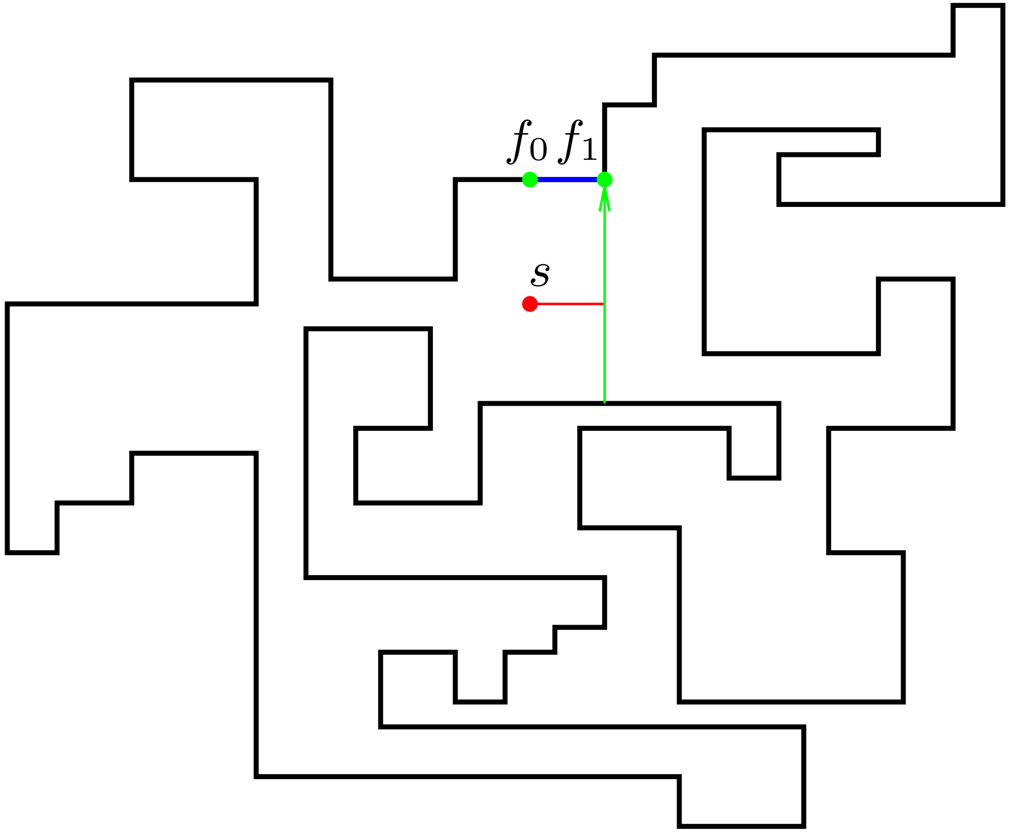
Exploring Rectilinear Polygons, cont'd

The strategy GO Deng, Kameda, Papadimitriou 1991/1998

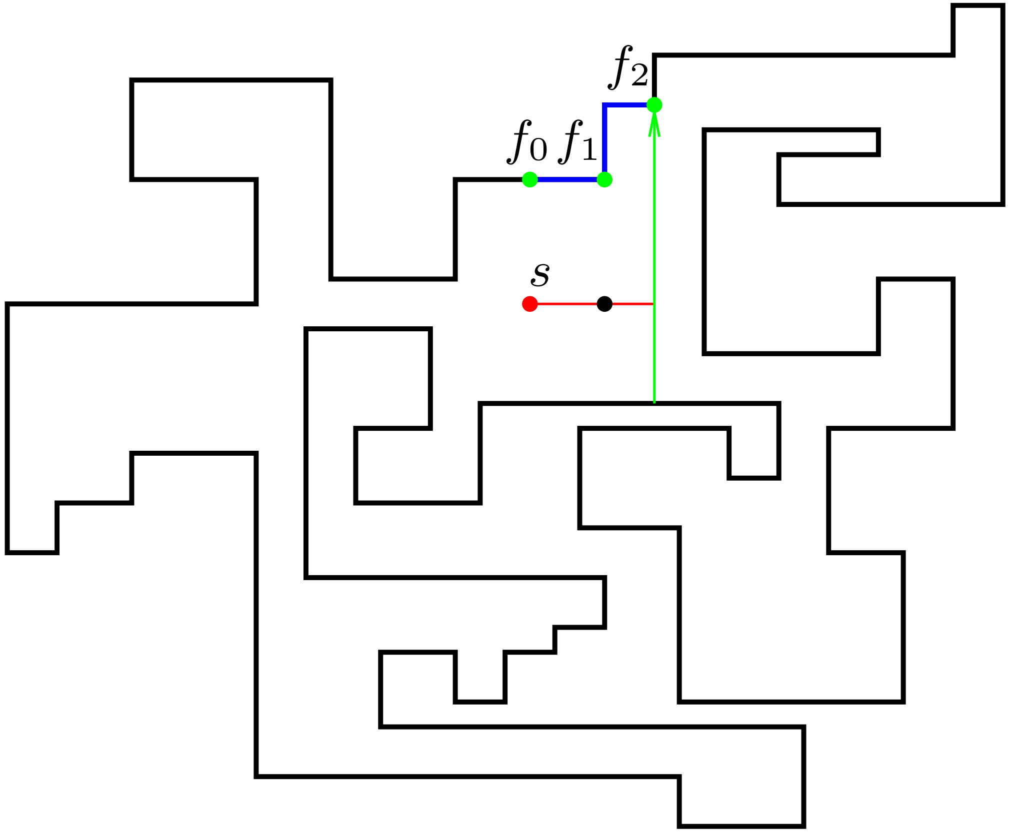
Strategy GO

1. Shoot a ray from the starting point upwards to the *principal projection point* on the boundary.
2. Scan the boundary clockwise until some portion of it (at the *frontier*) is not seen.
3. Move to the closest point on the associated (left) extension to the frontier.
4. Iterate from Step 2. until everything has been seen.

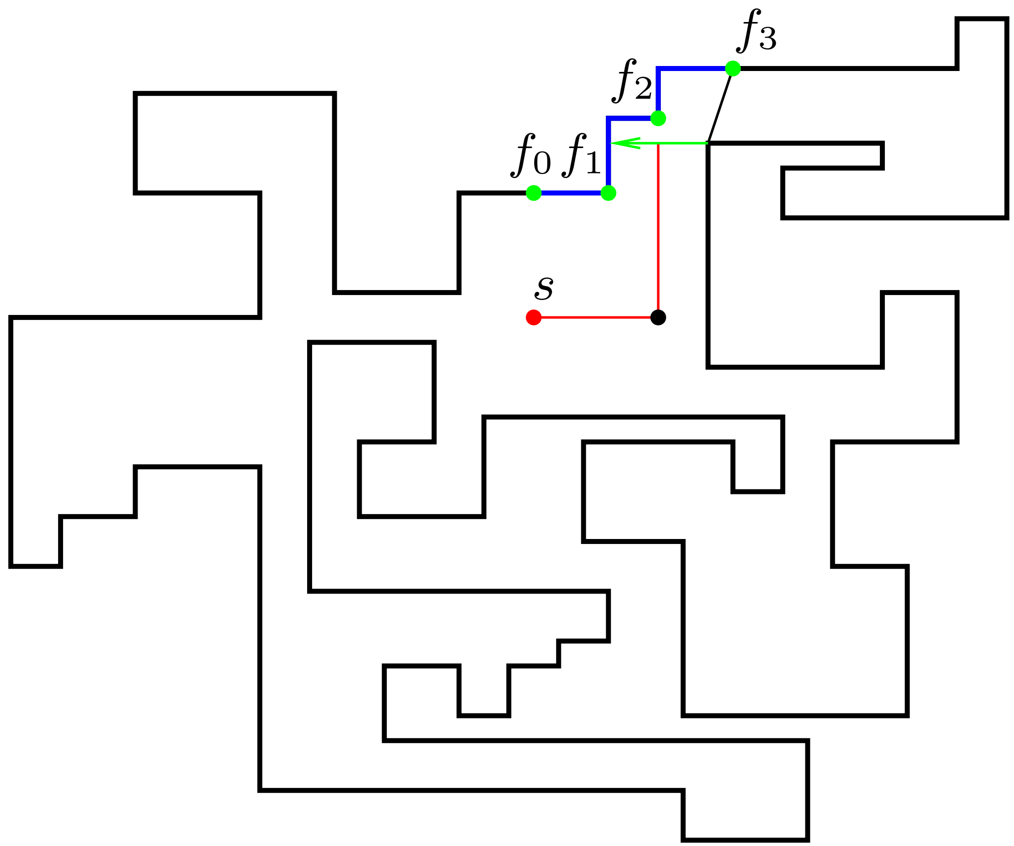
An example run of GO



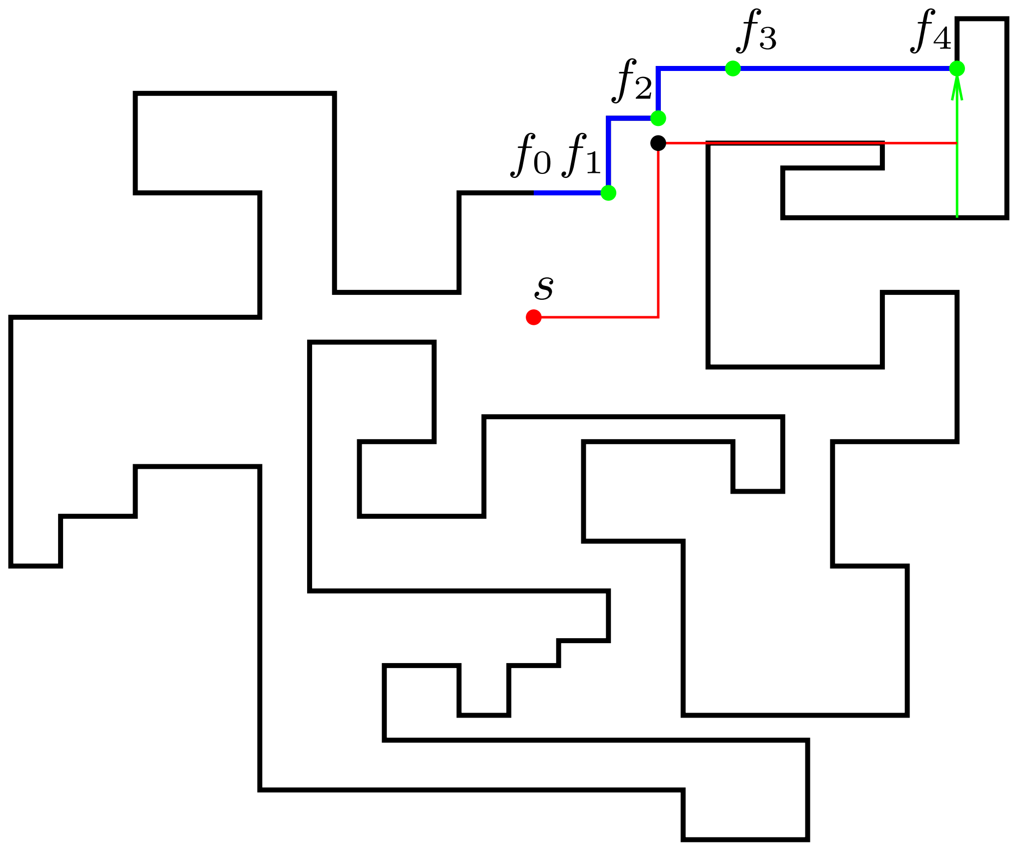
An example run of GO



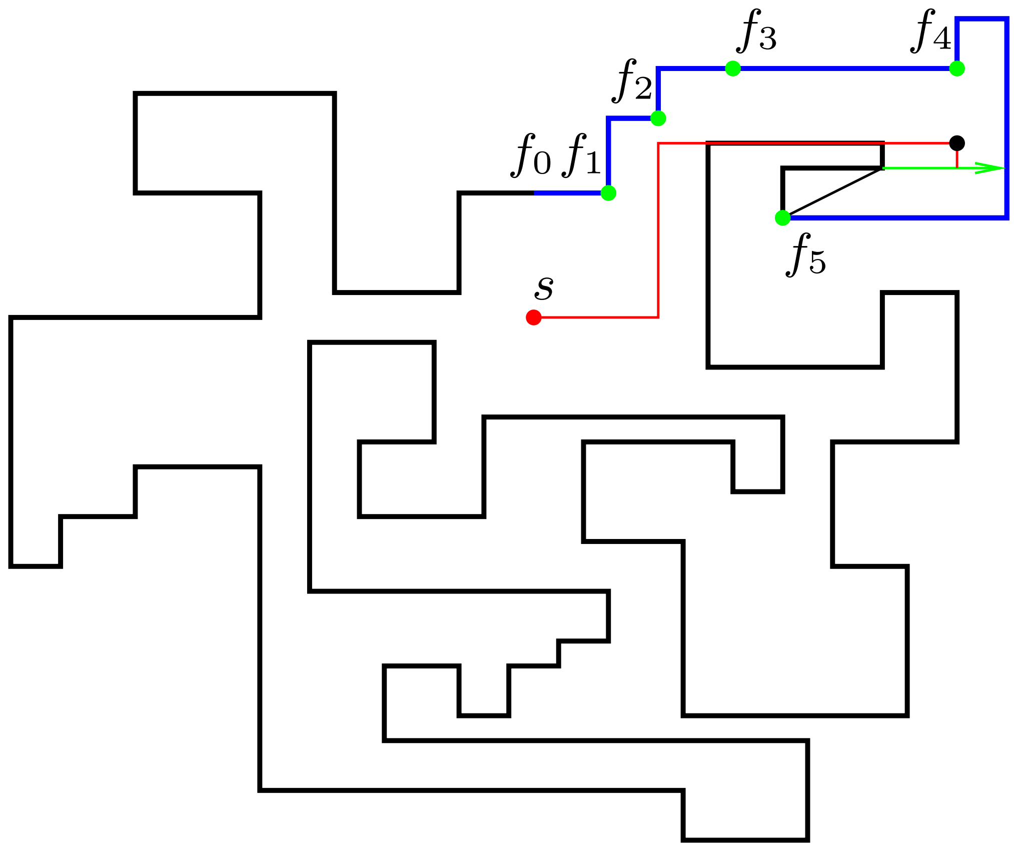
An example run of GO



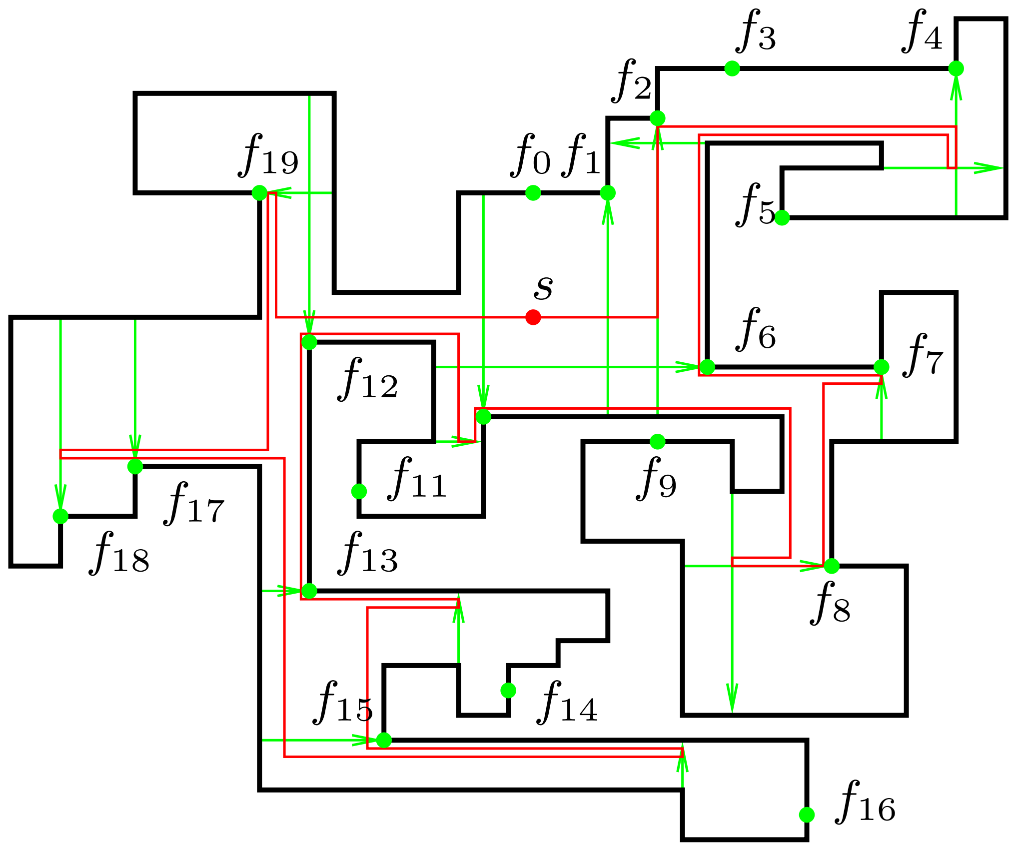
An example run of GO



An example run of GO



An example run of GO



Exploring Rectilinear Polygons, cont'd

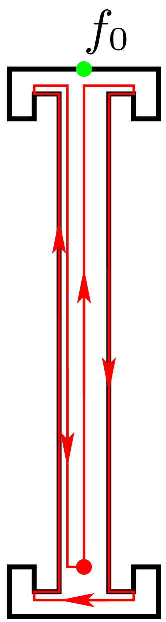
If the starting point is on the boundary (no need for step one, just scan), GO has optimal competitive factor 1

Why? Make a proof by induction on the number of extensions visited

If the starting point lies in the interior, GO has competitive ratio 2

Why? Every left extension has to be visited also by OPT

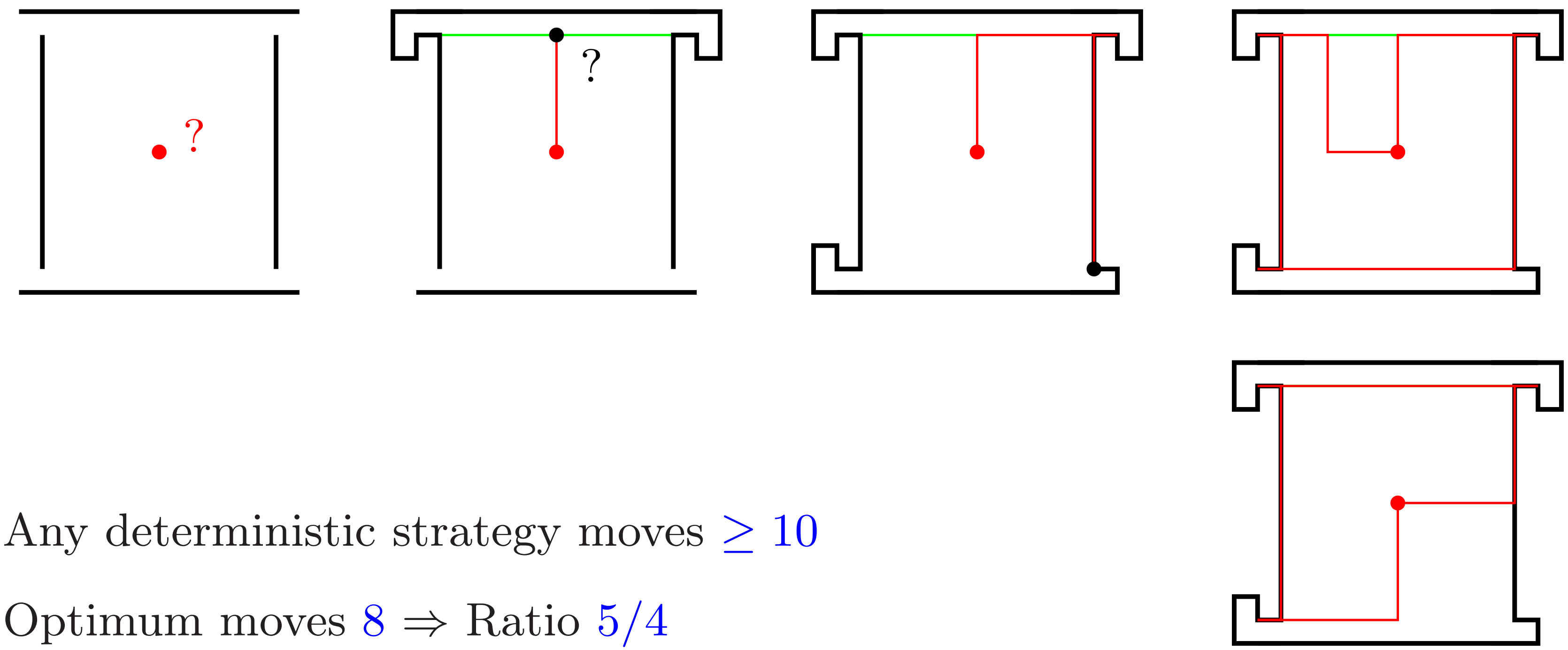
[Scanning clockwise or counterclockwise makes no difference]
We essentially have eight different strategies, dependent on four directions of principal projection point and two directions of scan



Exploring Rectilinear Polygons, cont'd

Proof by example (No joke!) Kleinberg 1994

Produce square of side length 2



Any deterministic strategy moves ≥ 10

Optimum moves $8 \Rightarrow$ Ratio $5/4$

Exploring Rectilinear Polygons, cont'd

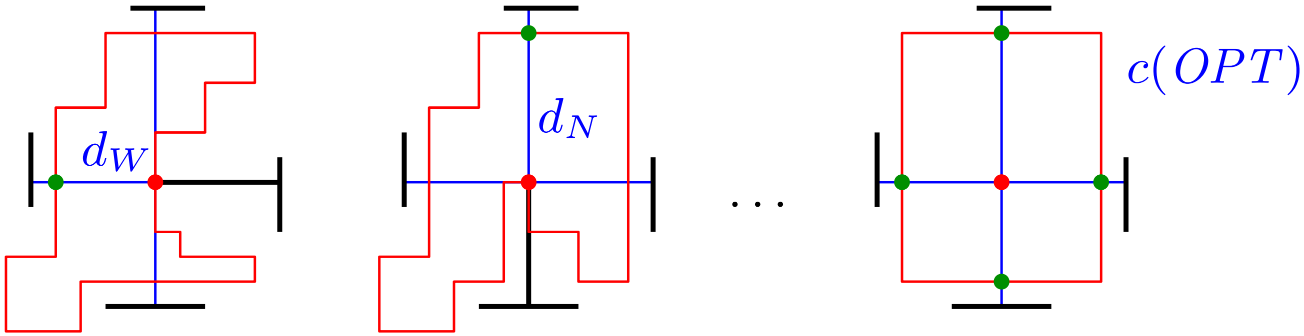
A Randomized Strategy Kleinberg 1994

- Choose a direction out of $\{N, E, S, W\}$ randomly, shoot a ray in this direction, and apply GO clockwise from there

Strategy uses two random bits

The strategy has expected competitive factor $5/4 = 1.25$

Exploring Rectilinear Polygons, cont'd

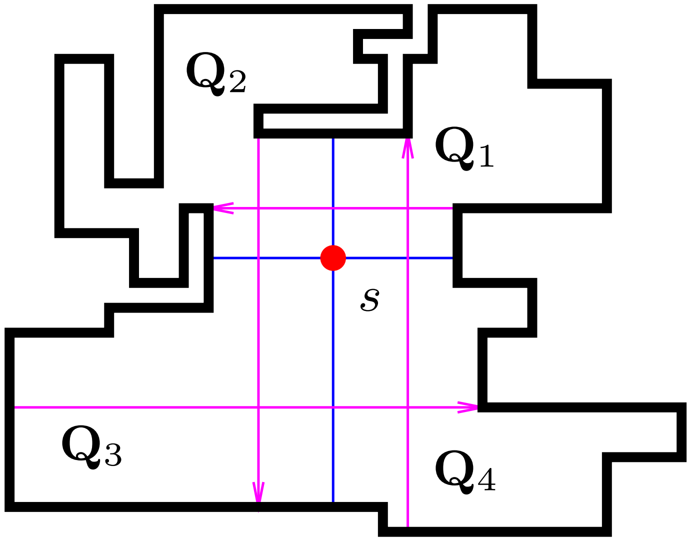


Argument: Introduce finger from each direction, optimal tour crosses opposite quadrant border at distance d_X, $X \in \{N, E, S, W\}$. $c(OPT)$ is at least $2(d_N + d_E + d_S + d_W)$, RGO tour is increased by $2d_X$ for randomly chosen X so

$$\frac{c(OPT) + \sum d_X/2}{c(OPT)} = 1 + \frac{\sum d_X/2}{c(OPT)} \leq 1 + \frac{\sum d_X/2}{2\sum d_X} = \frac{5}{4}$$

Exploring Rectilinear Polygons, cont'd

An Improved Deterministic Strategy, BGO Hammar *et al.* 2003



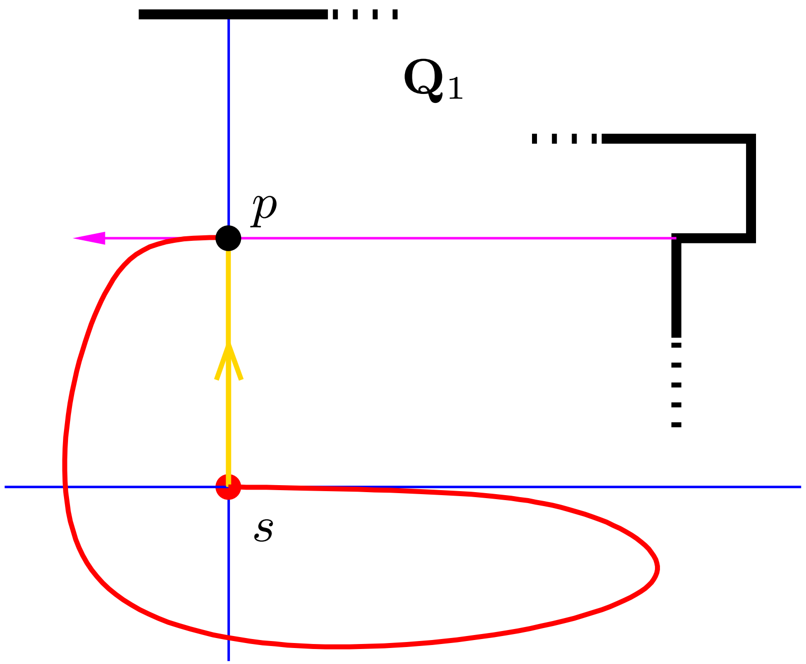
Divide the polygon into quadrants

Make initial scan to find the (left) extension furthest from s and do the exploration in the opposite direction using two frontiers, left and right

Move up until you have to make a decision

Exploring Rectilinear Polygons, cont'd

Case 0

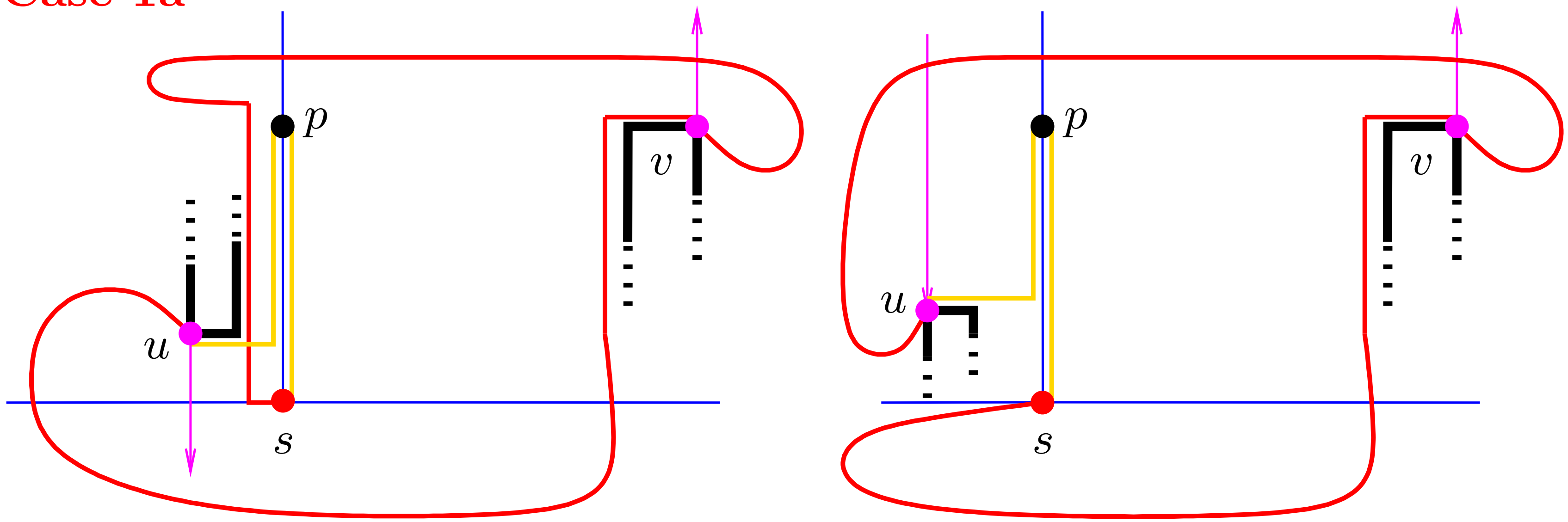


As motion proceeds up one quadrant becomes completely explored

$$R_{\mathrm{BGO}_0} = 1$$

Exploring Rectilinear Polygons, cont'd

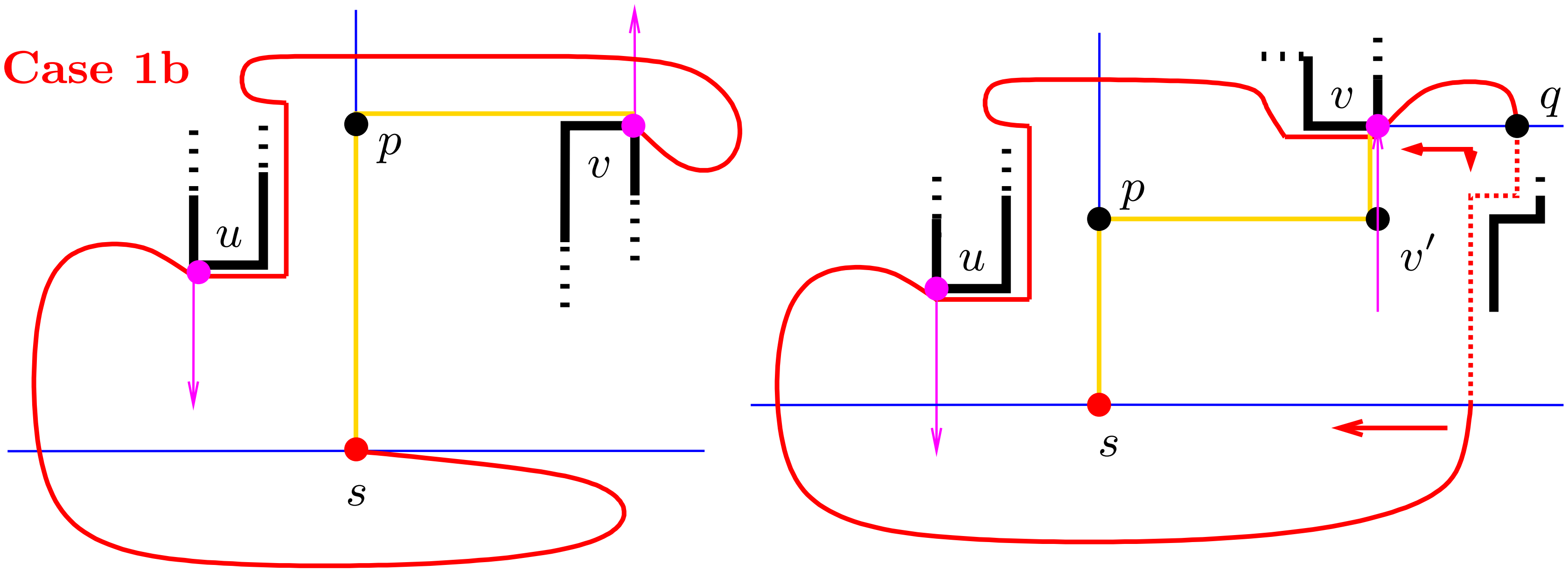
Case 1a



$$||s,p||_y + ||s,u||_x \leq ||s,v||_x$$

$$R_{\mathrm{BGO}_{1a}} = 3/2.$$

Exploring Rectilinear Polygons, cont'd



$$||s,p||_y + ||s,u||_x > ||s,v||_x$$

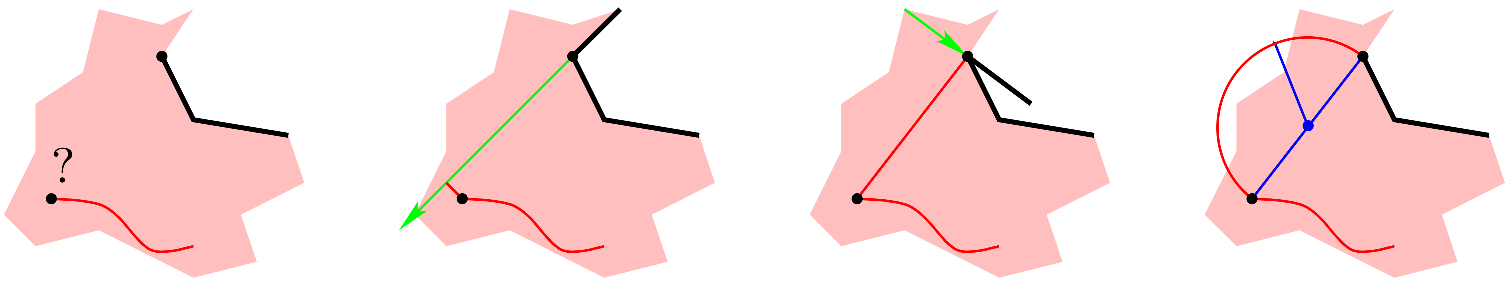
$$R_{\text{BGO}_{1b}} = 3/2.$$

Case 2 Comprises three further subcases handled similarly

$$R_{\text{BGO}_2} = 3/2$$

Exploring Simple Polygons

Problem: Not clear how to go to an extension

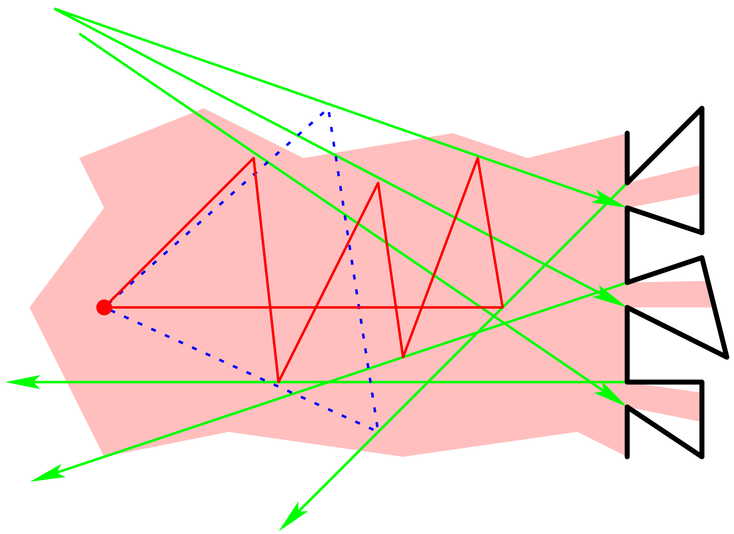


Follow rectilinear path + 45° when needed to reach extension, like in streets. Detour is $1+\sqrt{2} \approx 2.41$

Better solution follow half circle centered on midpoint. Detour is $\frac{\pi}{2} \approx 1.57$

Exploring Simple Polygons

Problem: Not even clear what order to explore/visit windows



In order along the boundary can make tour zig-zag very much

$\Rightarrow$ no competitive bound

Solution: Group similar unexplored windows together

Strategy with competitive factor ≤ 26.5 **Messy!!!**

Hoffman, Icking, Klein, Kriegel 1997

Multi-Agent Tree Exploration

Problem: k point agents $A_1, \ldots, A_k$ is placed at the root of a tree and are required to explore it, i.e., make certain every node is visited by some agent A_i, each agent returns to root.

Every A_i has identification system, memory, communication system

What is the best exploration strategy for the agent?

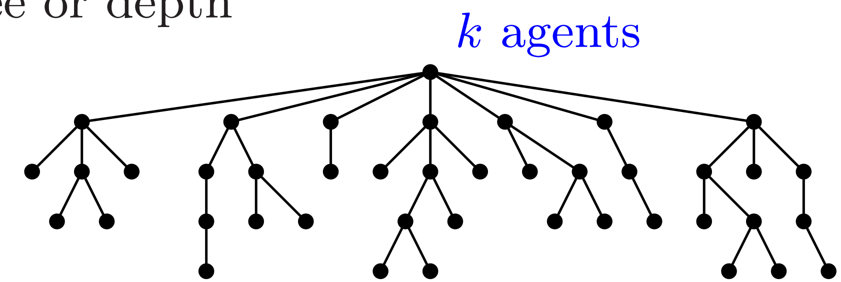
We compare length of A_i's tour that moves the longest distance in the strategy to length of A_j's tour that moves the longest distance in the optimal solution **(Makespan)**

Multi-Agent Tree Exploration, cont'd

All k agents start at the root.

Agents in nodes are aware of edges leading to child nodes but nothing more

No limits on degree or depth



$k = 1$: Preorder traversal is optimal.

In general, **computing** the optimal tours is **NP-complete** but...

$$\max\{\frac{\mathit{Traversal}}{k}, 2 \cdot \mathit{Depth}\} \leq \text{optimal tour} \leq \frac{\mathit{Traversal}}{k} + 2 \cdot \mathit{Depth}$$

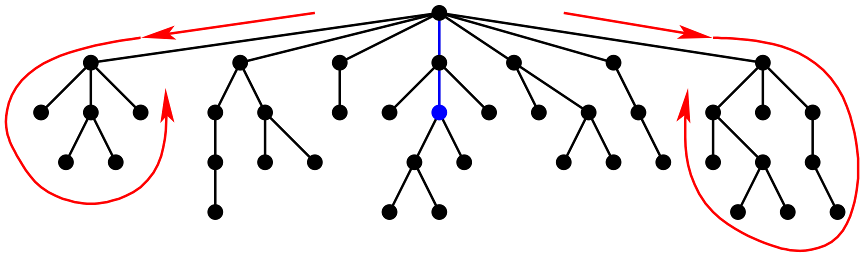
Good approximation **2**

Multi-Agent Tree Exploration, cont'd

$k = 2$

Strategy:

Do preorder traversal with one and reverse preorder traversal with the other.
When they meet and have seen all nodes together, go back to root



Total $\mathit{Traversal} = 2 \cdot \mathit{Edges}$

Each agent has visited $\leq 2 \cdot \mathit{Edges}/2$ before they meet, then $\leq \mathit{Depth}$ to go to the root

$$\frac{2 \cdot \mathit{Edges}/2 + \mathit{Depth}}{\text{optimal tour}} \leq \frac{2 \cdot \mathit{Edges}/2 + \mathit{Depth}}{\max\{\mathit{Edges}, 2 \cdot \mathit{Depth}\}} \leq \frac{3}{2}$$

Multi-Agent Tree Exploration, cont'd

Upper Bound (Fraigniaud, Gasieniec, Kowalski, Pelc 2006).

Simple and natural strategy: (works in synchronous steps)

Strategy Collective Exploration

Consider agents in a node v, in a step perform:

if (not all subtrees of v are explored) {

Distribute agents at v evenly among unexplored subtrees

} **else** /*all subtrees of v are explored*/ {

if (no subtree of v contains agents) {

Move agents at v to the parent of v

} **else** /*some subtree of v contains agents*/ Wait

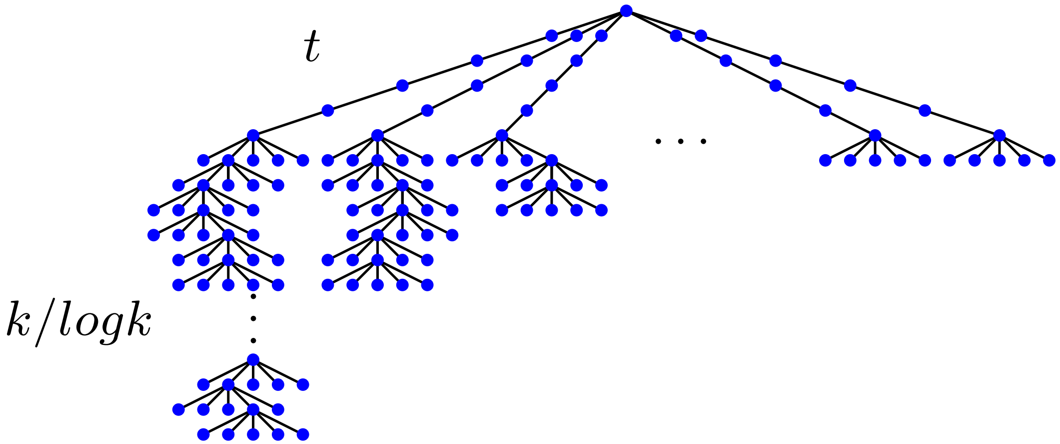
}

Wait is important for the analysis. Competitive ratio is $O(k/\log k)$

Multi-Agent Tree Exploration, cont'd

Lower Bound (Dynia, Lopuszanski, Schindelhauer 2007)

Jellyfish tree: ($t \geq k$)



Keep fraction $1/i$ of the tentacles in level i. Tree has size $O(tk)$

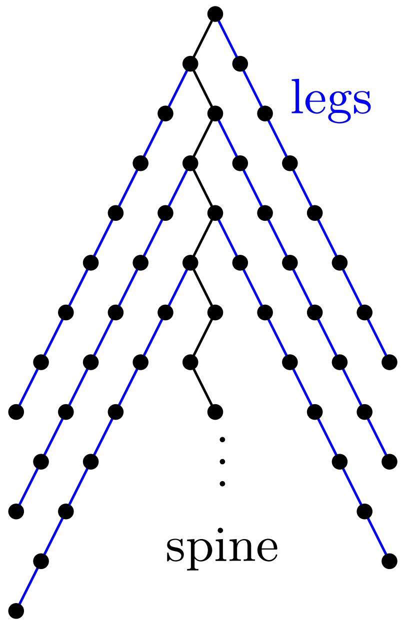
Any strategy must use $t \log k / \log \log k$ steps, optimum uses $O(t)$

$\Rightarrow$ competitive ratio $\Omega(\log k / \log \log k)$

Multi-Agent Tree Exploration, cont'd

Lower Bound for Monotone Strategies

 produces



Any strategy will only come $\log k$ steps down the spine before it has to wait for legs to be explored

Optimum comes k steps down $\Rightarrow$ competitive ratio $\dfrac{k}{\log k}$

Conclusions

- Competitive analysis
- List access and paging
- Variants on linear search
- Geometric searching and exploration (polygons)
- Exploring trees with multiple agents

Thank you for listening

Questions? **Comments?**