Optimal Exploration of Terrains with Obstacles

Jurek Czyzowicz[1] David Ilcinkas[2]
Arnaud Labourel[1,2] Andrzej Pelc[1]

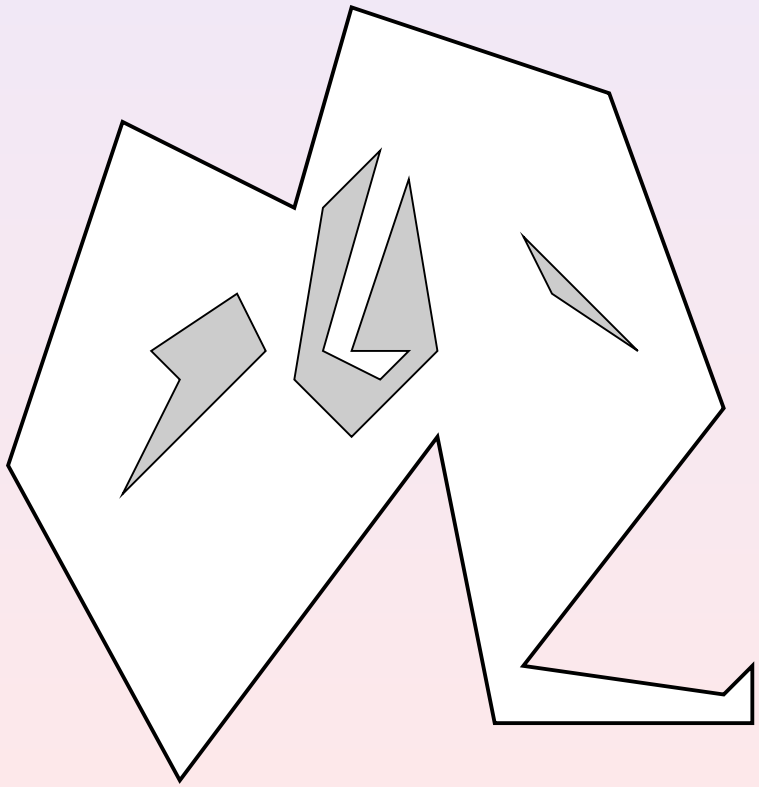
[1]Université du Québec en Outaouais, Canada
[2]CNRS & Univ. de Bordeaux, France

Séminaire "Algorithmique distribue", LaBRI
May 3, 2010

The problem

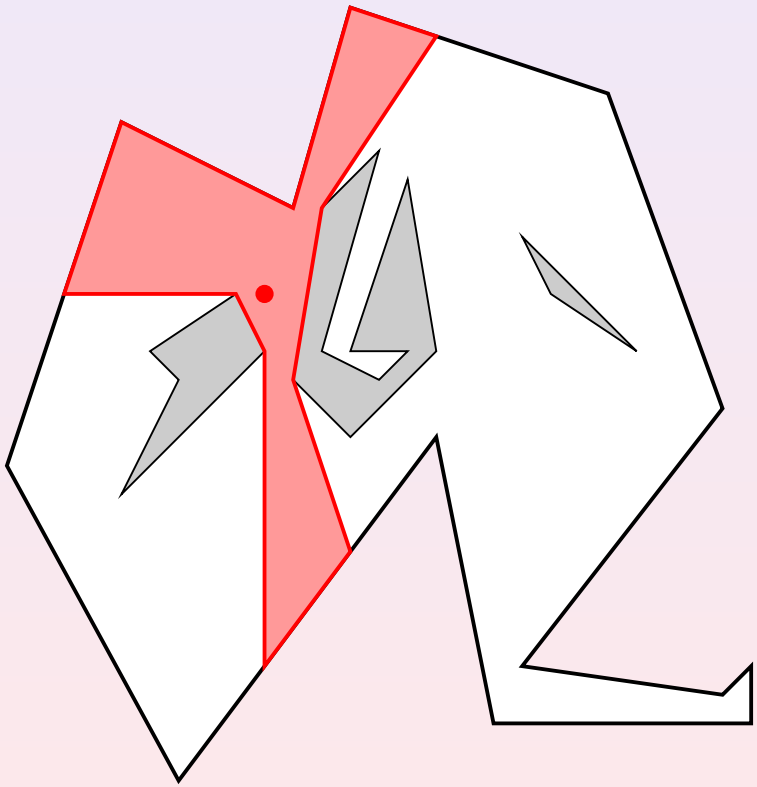
Exploration of terrains with obstacles

A mobile robot has to explore/see all points of an unknown bounded terrain, possibly with obstacles.



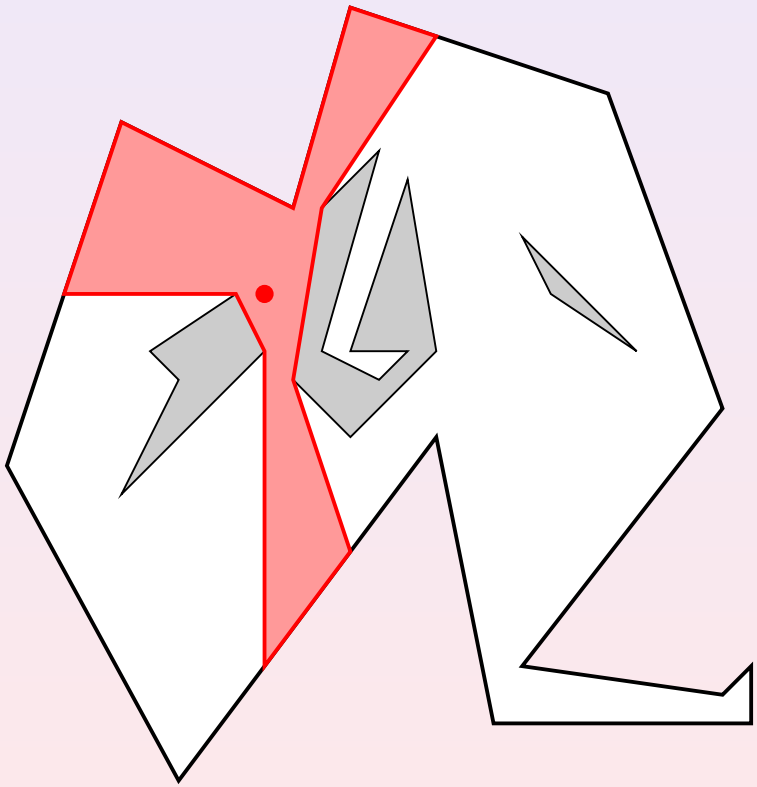
Vision

Unlimited vision

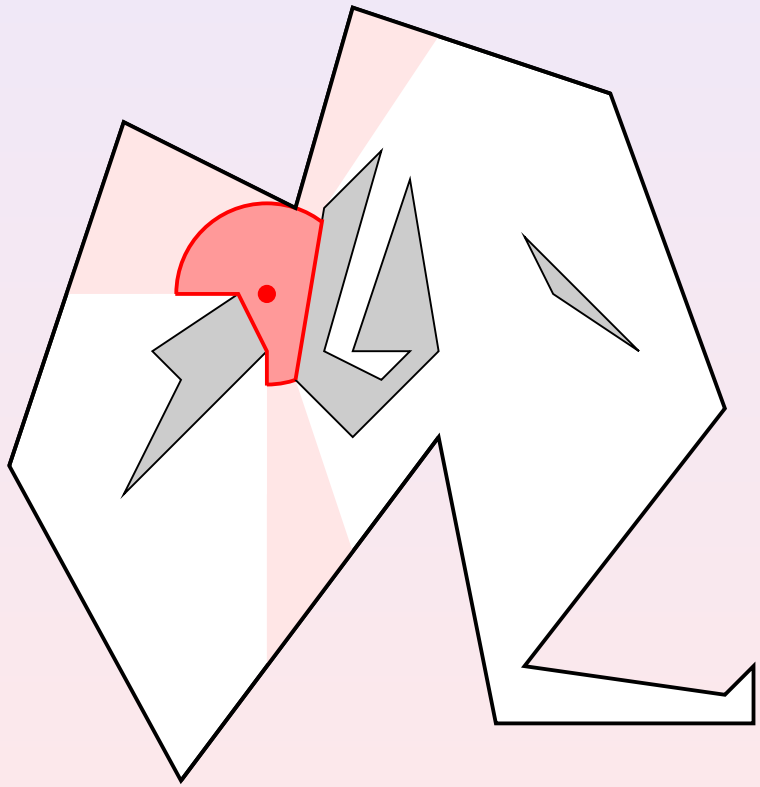


Vision

Unlimited vision



Limited vision



Motivations

Motivations / Applications

Exploration / searching / mapping of

- unsafe environments
- unreachable environments

Related (off-line) problems

- with unlimited vision
  - Gallery tour problem (NP-hard)
  - Watchman's route (polynomial)
- with limited vision
  - Lawn mowing problem (NP-hard)
  - Pocket milling problem (NP-hard)

Motivations

Motivations / Applications

Exploration / searching / mapping of

- unsafe environments
- unreachable environments

Related (off-line) problems

- with unlimited vision
  - Gallery tour problem (NP-hard)
  - Watchman's route (polynomial)
- with limited vision
  - Lawn mowing problem (NP-hard)
  - Pocket milling problem (NP-hard)

Motivations

Motivations / Applications

Exploration / searching / mapping of

- unsafe environments
- unreachable environments

Related (off-line) problems

- with unlimited vision
  - Gallery tour problem (NP-hard)
  - Watchman's route (polynomial)
- with limited vision
  - Lawn mowing problem (NP-hard)
  - Pocket milling problem (NP-hard)

Performance measures

Algorithm $\mathcal{A}$ having no a priori knowledge on the terrain.

Worst-case complexity

Worst-case length of $\mathcal{A}$'s trajectory in the family of terrains of bounded parameters

- P: total perimeter
- A: area of the terrain
- k: number of obstacles

Competitive ratio

$$\max_{\mathcal{T}} \frac{\text{length of } \mathcal{A}\text{'s trajectory}}{\text{length of an optimal algorithm knowing the terrain } \mathcal{T}}$$

Performance measures

Algorithm $\mathcal{A}$ having no a priori knowledge on the terrain.

Worst-case complexity

Worst-case length of $\mathcal{A}$'s trajectory in the family of terrains of bounded parameters

- P: total perimeter
- A: area of the terrain
- k: number of obstacles

Competitive ratio

$$\max_{\mathcal{T}} \frac{\text{length of } \mathcal{A}\text{'s trajectory}}{\text{length of an optimal algorithm knowing the terrain } \mathcal{T}}$$

Performance measures

Algorithm $\mathcal{A}$ having no a priori knowledge on the terrain.

Worst-case complexity

Worst-case length of $\mathcal{A}$'s trajectory in the family of terrains of bounded parameters

- P: total perimeter
- A: area of the terrain
- k: number of obstacles

Competitive ratio

$$\max_{\mathcal{T}} \frac{\text{length of } \mathcal{A}\text{'s trajectory}}{\text{length of an optimal algorithm knowing the terrain } \mathcal{T}}$$

Related work (unlimited vision)

Rectilinear polygon without obstacles

- [Deng, Kameda, Papadimitriou, 1998]: competitive ratio 2
- [Hammar, Nilsson, Schuierer, 2002]: competitive ratio $\frac{5}{3}$
- [Hammar, Nilsson, Persson, 2006]: competitive ratio $\frac{3}{2}$
- [Kleinberg, 1994]: (randomized) competitive ratio $\frac{5}{4}$
  no deterministic algorithm with competitive ratio $< \frac{5}{4}$

Polygonal terrain with obstacles

- [Kalyanasundaram, Pruhs, 1993]: competitive ratio $O(k)$
  ($O(\sqrt{k})$ in some special cases)
- [Deng, Kameda, Papadimitriou, 1998]: competitive ratio $\omega(1)$
- [Albers, Kursawe, Schuierer, 2002]: competitive ratio $\Omega(\sqrt{k})$

Related work (unlimited vision)

Rectilinear polygon without obstacles

- [Deng, Kameda, Papadimitriou, 1998]: competitive ratio 2
- [Hammar, Nilsson, Schuierer, 2002]: competitive ratio $\frac{5}{3}$
- [Hammar, Nilsson, Persson, 2006]: competitive ratio $\frac{3}{2}$
- [Kleinberg, 1994]: (randomized) competitive ratio $\frac{5}{4}$
  no deterministic algorithm with competitive ratio $< \frac{5}{4}$

Polygonal terrain with obstacles

- [Kalyanasundaram, Pruhs, 1993]: competitive ratio $O(k)$
  ($O(\sqrt{k})$ in some special cases)
- [Deng, Kameda, Papadimitriou, 1998]: competitive ratio $\omega(1)$
- [Albers, Kursawe, Schuierer, 2002]: competitive ratio $\Omega(\sqrt{k})$

Related work (limited vision)

- Y. Gabriely, E. Rimon, Int. Conf. of Robotics and Automaton (ICRA), 2001.
- Y. Gabriely, E. Rimon, Computational Geometry: Theory and Applications, 2003.
- S.K. Ghosh, J.W. Burdick, A. Bhattacharya, S. Sarkar, Robotics & Automation Magazine, 2008.
- C. Icking, T. Kamphans, R. Klein, E. Langetepe, European Workshop on Computational Geometry, 2000.
- A. Kolenderska, A. Kosowski, M. Malafiejski, P. Zylinski, SIROCCO, 2009.
- S. Ntafos, Comput. Geom. Theory Appl., 1992.

Our results

We consider the worst-case trajectory length measure.

- P: total perimeter
- D: diameter of the (convex hull of the) terrain
- A: area of the terrain
- k: number of obstacles

Unlimited vision

- $\Omega(P + D\sqrt{k})$, even if the terrain is a priori known
- $O(P + D\sqrt{k})$, constructive proof (algorithm)

Limited vision

- $\Omega(P + A + \sqrt{Ak})$, even if the terrain is a priori known
- $O(P + A + \sqrt{Ak})$, if k or A is known (algorithm)
- $O(\min\{\log A, \log k\}(P + A + \sqrt{Ak}))$, nothing known

Our results

We consider the worst-case trajectory length measure.

- P: total perimeter
- D: diameter of the (convex hull of the) terrain
- A: area of the terrain
- k: number of obstacles

Unlimited vision

- $\Omega(P + D\sqrt{k})$, even if the terrain is a priori known
- $O(P + D\sqrt{k})$, constructive proof (algorithm)

Limited vision

- $\Omega(P + A + \sqrt{Ak})$, even if the terrain is a priori known
- $O(P + A + \sqrt{Ak})$, if k or A is known (algorithm)
- $O(\min\{\log A, \log k\}(P + A + \sqrt{Ak}))$, nothing known

Our results

We consider the worst-case trajectory length measure.

- P: total perimeter
- D: diameter of the (convex hull of the) terrain
- A: area of the terrain
- k: number of obstacles

Unlimited vision

- $\Omega(P + D\sqrt{k})$, even if the terrain is a priori known
- $O(P + D\sqrt{k})$, constructive proof (algorithm)

Limited vision

- $\Omega(P + A + \sqrt{Ak})$, even if the terrain is a priori known
- $O(P + A + \sqrt{Ak})$, if k or A is known (algorithm)
- $O(\min\{\log A, \log k\}(P + A + \sqrt{Ak}))$, nothing known

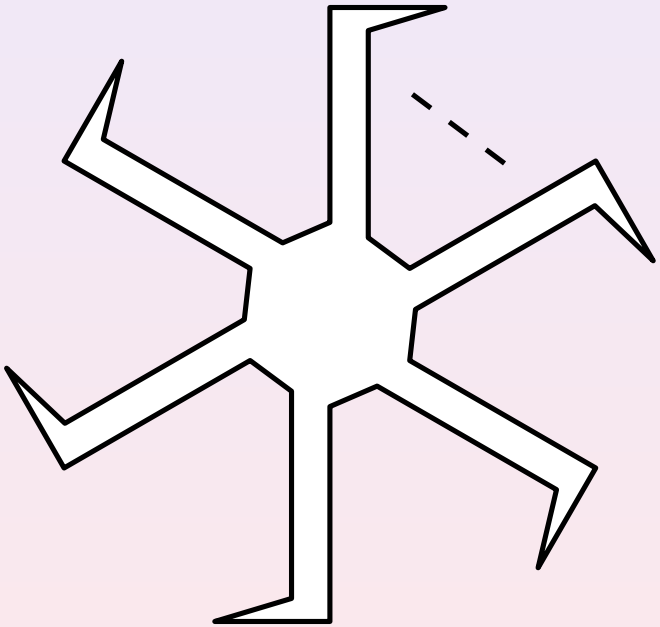
Outline

Lower bound

Lower bound $\Omega(P + D\sqrt{k})$, even if the terrain is known.
(Reminder: P = perimeter; D = diameter; k = # obstacles)

Lower bound

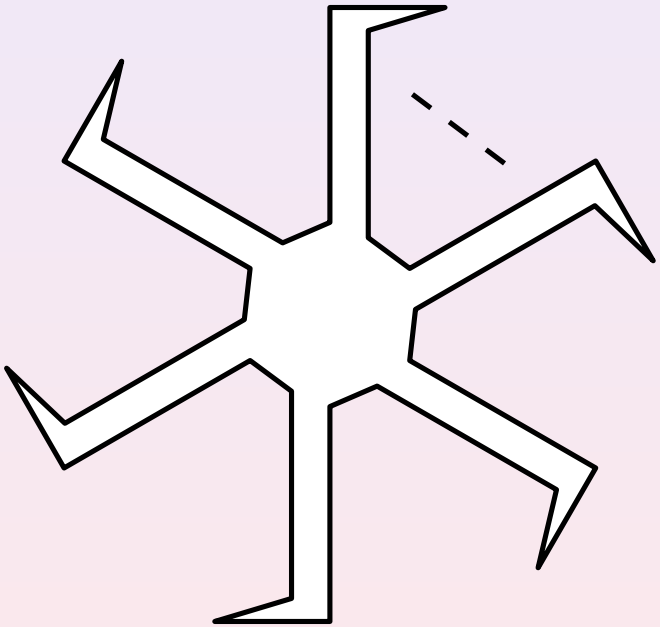
Lower bound $\Omega(P + D\sqrt{k})$, even if the terrain is known.
(Reminder: P = perimeter; D = diameter; k = # obstacles)



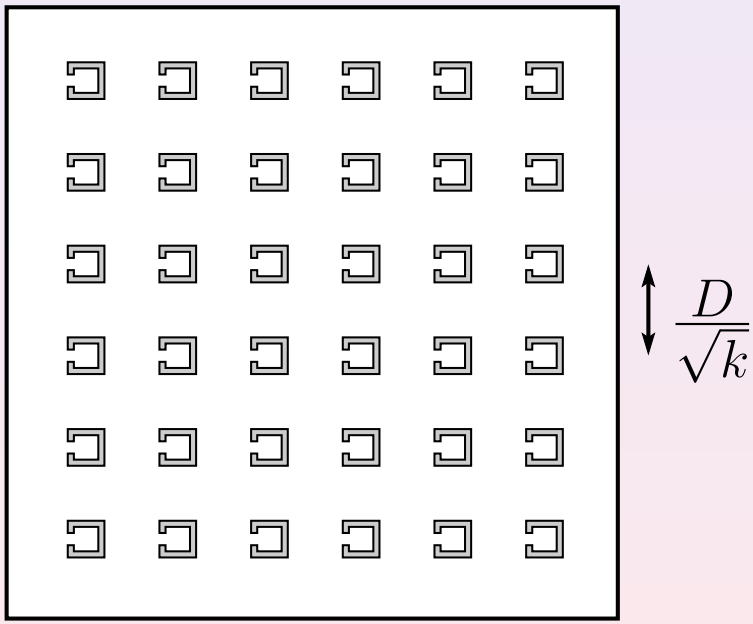
Lower bound $\Omega(P)$

Lower bound

Lower bound $\Omega(P + D\sqrt{k})$, even if the terrain is known.
(Reminder: P = perimeter; D = diameter; k = # obstacles)



Lower bound $\Omega(P)$



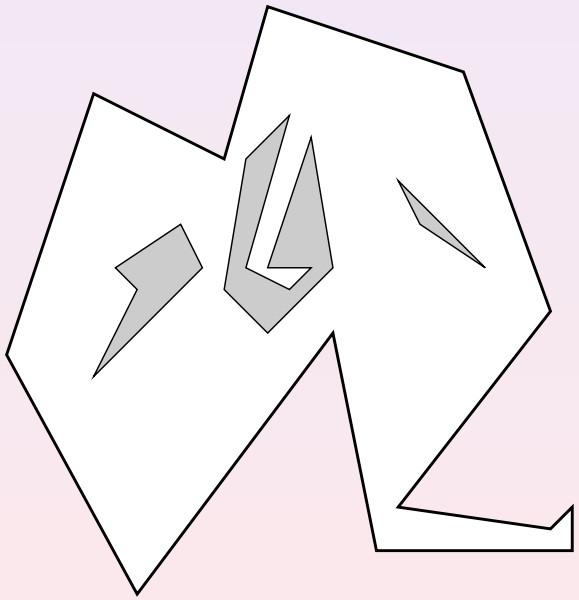
Lower bound $\Omega(D\sqrt{k})$

Upper bound

Upper bound $O(P + D \cdot k)$, without a priori knowledge.
(Reminder: P = perimeter; D = diameter; k = # obstacles)

Naive algorithm

- Reach the boundary
- Follow/explore the boundary
- **While** unvisited obstacles
  - Go to an unvisited obstacle
  - Go around it

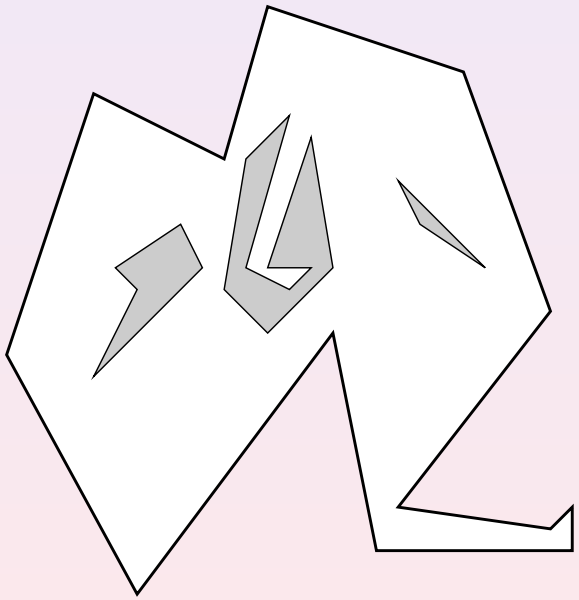


Upper bound

Upper bound $O(P + D\sqrt{k})$, without a priori knowledge.
(Reminder: P = perimeter; D = diameter; k = # obstacles)

Our algorithm

- Reach the boundary
- Follow/explore the boundary
- Initialize the quadtree partition
- **While** unvisited obstacles
  - Go to a close unvisited obstacle
  - Go around it
  - Refine the quadtree partition

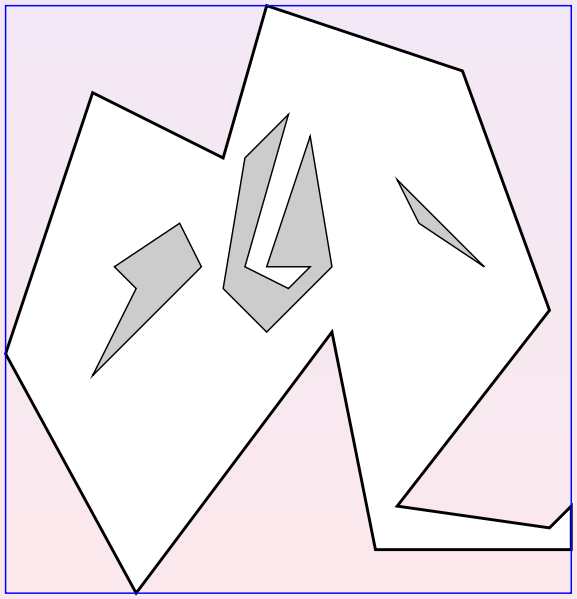


Upper bound

Upper bound $O(P + D\sqrt{k})$, without a priori knowledge.
(Reminder: P = perimeter; D = diameter; k = # obstacles)

Our algorithm

- Reach the boundary
- Follow/explore the boundary
- Initialize the quadtree partition
- **While** unvisited obstacles
  - Go to a close unvisited obstacle
  - Go around it
  - Refine the quadtree partition

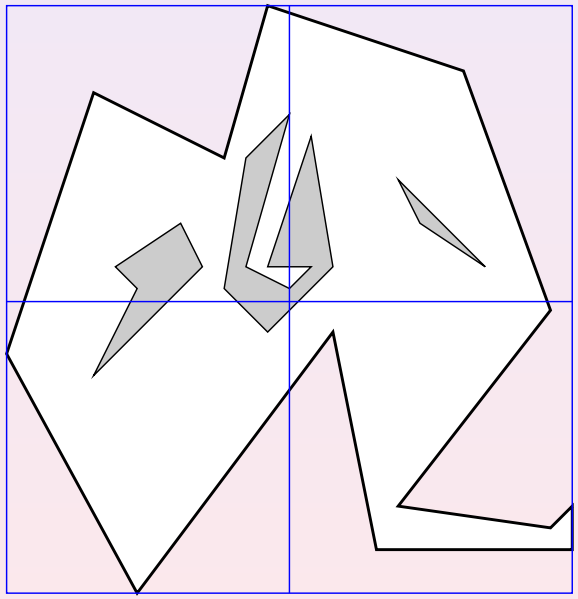


Upper bound

Upper bound $O(P + D\sqrt{k})$, without a priori knowledge.
(Reminder: P = perimeter; D = diameter; k = # obstacles)

Our algorithm

- Reach the boundary
- Follow/explore the boundary
- Initialize the quadtree partition
- **While** unvisited obstacles
  - Go to a close unvisited obstacle
  - Go around it
  - Refine the quadtree partition

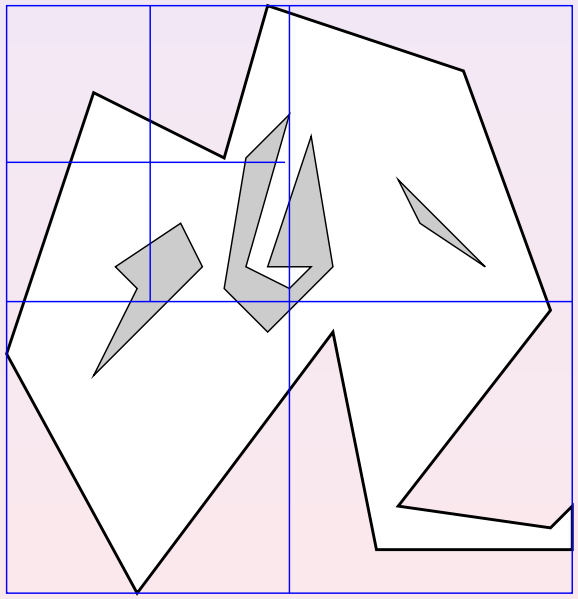


Upper bound

Upper bound $O(P + D\sqrt{k})$, without a priori knowledge.
(Reminder: P = perimeter; D = diameter; k = # obstacles)

Our algorithm

- Reach the boundary
- Follow/explore the boundary
- Initialize the quadtree partition
- **While** unvisited obstacles
  - Go to a close unvisited obstacle
  - Go around it
  - Refine the quadtree partition

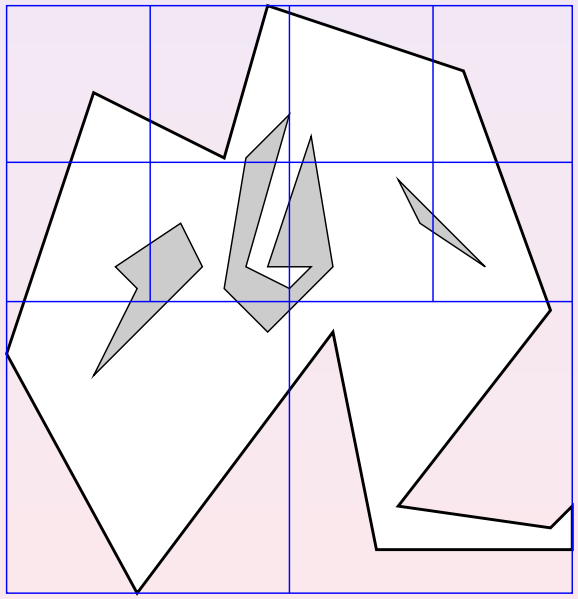


Upper bound

Upper bound $O(P + D\sqrt{k})$, without a priori knowledge.
(Reminder: P = perimeter; D = diameter; k = # obstacles)

Our algorithm

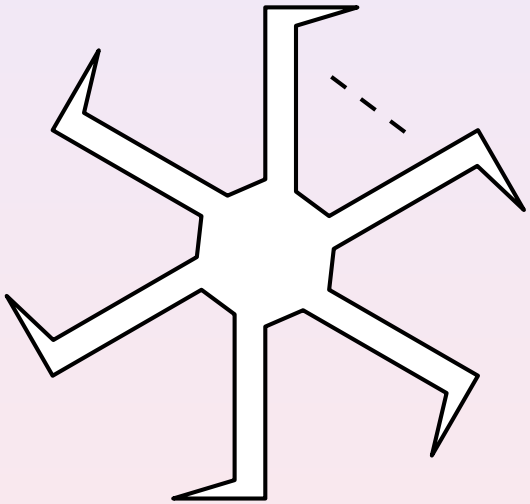
- Reach the boundary
- Follow/explore the boundary
- Initialize the quadtree partition
- **While** unvisited obstacles
  - Go to a close unvisited obstacle
  - Go around it
  - Refine the quadtree partition



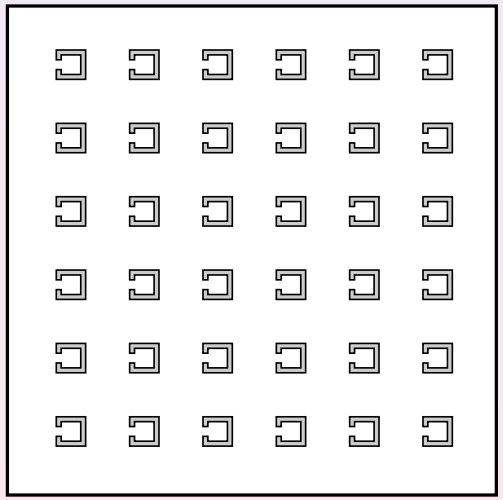
Outline

Lower bound

Lower bound $\Omega(P + A + \sqrt{A \cdot k})$, even if the terrain is known.
(Reminder: P = perimeter; A = area; k = # obstacles)



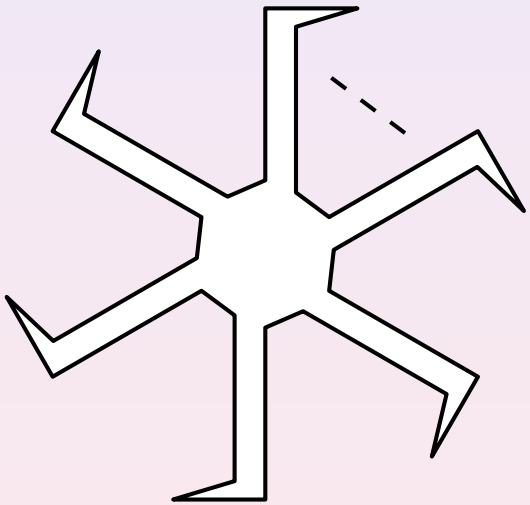
Lower bound
$\Omega(P)$



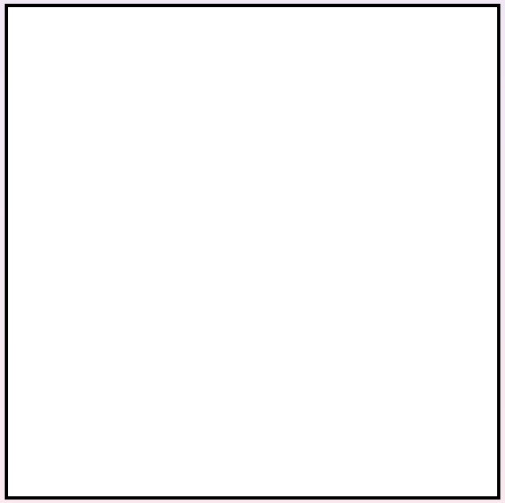
Lower bound
$\Omega(\sqrt{A \cdot k})$

Lower bound

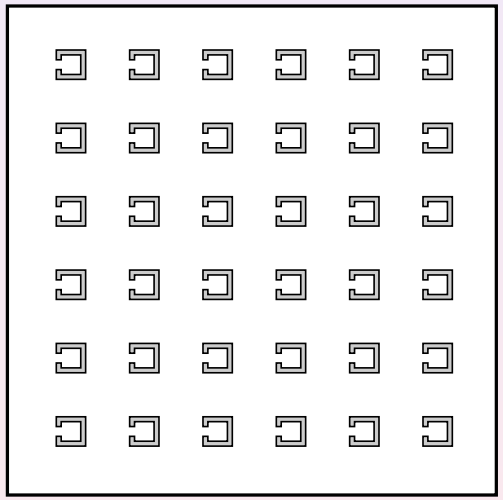
Lower bound $\Omega(P + A + \sqrt{A \cdot k})$, even if the terrain is known.
(Reminder: P = perimeter; A = area; k = # obstacles)



Lower bound
$\Omega(P)$



Lower bound
$\Omega(A)$



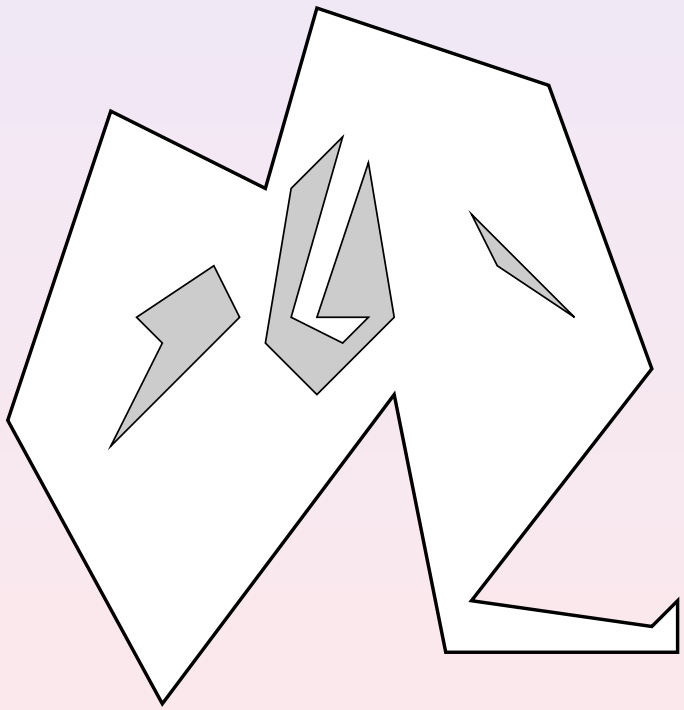
Lower bound
$\Omega(\sqrt{A \cdot k})$

Upper bound (1)

Basic idea: use the unlimited vision algorithm

Algorithm LET(F)

- Reach the boundary
- Follow/explore the boundary
- Partition the terrain in cells (of side $F \leq 1$)
- $\forall$ cell in DFS order
  - Explore the cell

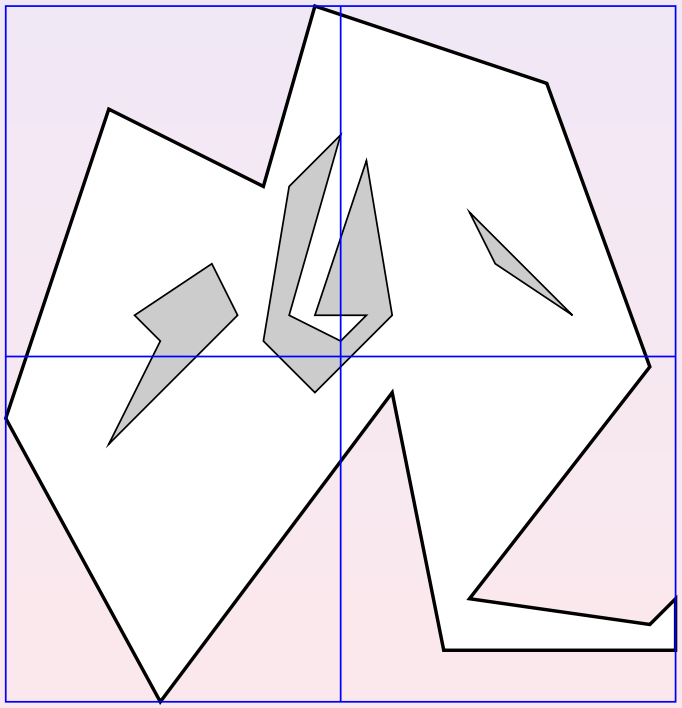


Upper bound (1)

Basic idea: use the unlimited vision algorithm

Algorithm LET(F)

- Reach the boundary
- Follow/explore the boundary
- Partition the terrain in cells (of side $F \leq 1$)
- $\forall$ cell in DFS order
  - Explore the cell

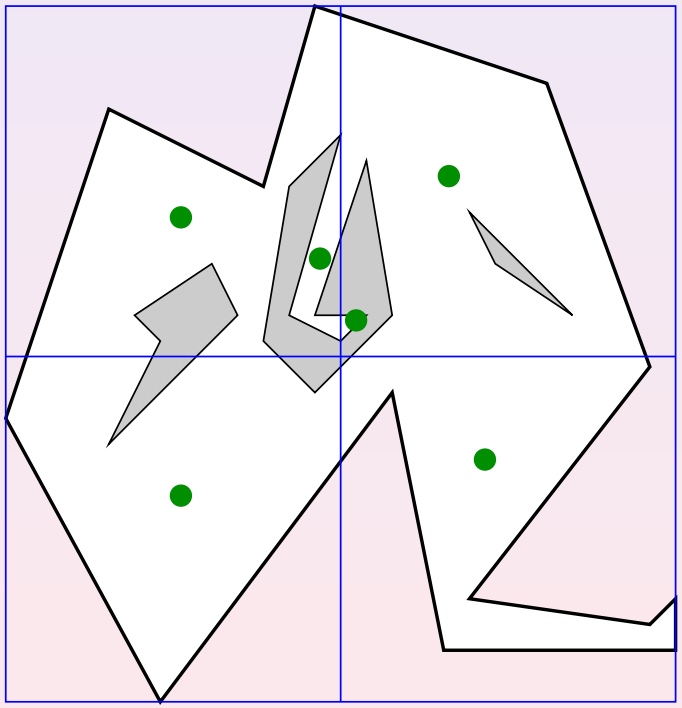


Upper bound (1)

Basic idea: use the unlimited vision algorithm

Algorithm LET(F)

- Reach the boundary
- Follow/explore the boundary
- Partition the terrain in cells (of side $F \leq 1$)
- $\forall$ cell in DFS order
  - Explore the cell

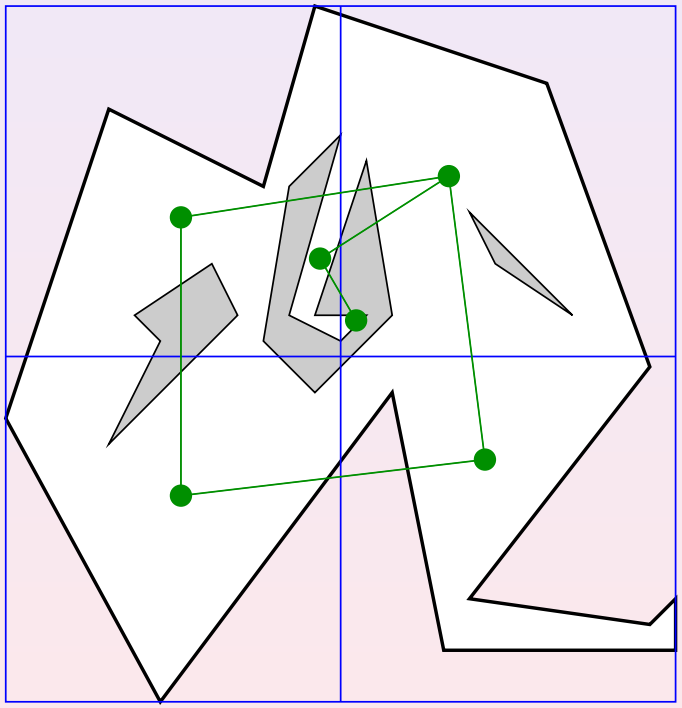


Upper bound (1)

Basic idea: use the unlimited vision algorithm

Algorithm LET(F)

- Reach the boundary
- Follow/explore the boundary
- Partition the terrain in cells (of side $F \leq 1$)
- $\forall$ cell in DFS order
  - Explore the cell

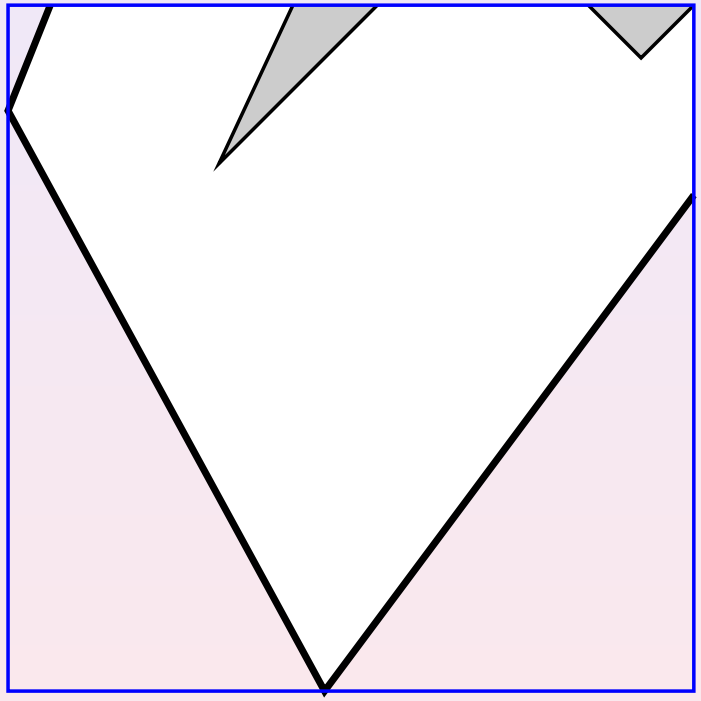


Cell exploration

Basic idea: use the unlimited vision naive algorithm

Naive algorithm

- Reach the boundary of cell c
- Follow/explore the boundary
- **While** unvisited obstacles
  - Go to an unvisited obstacle
  - Go around it



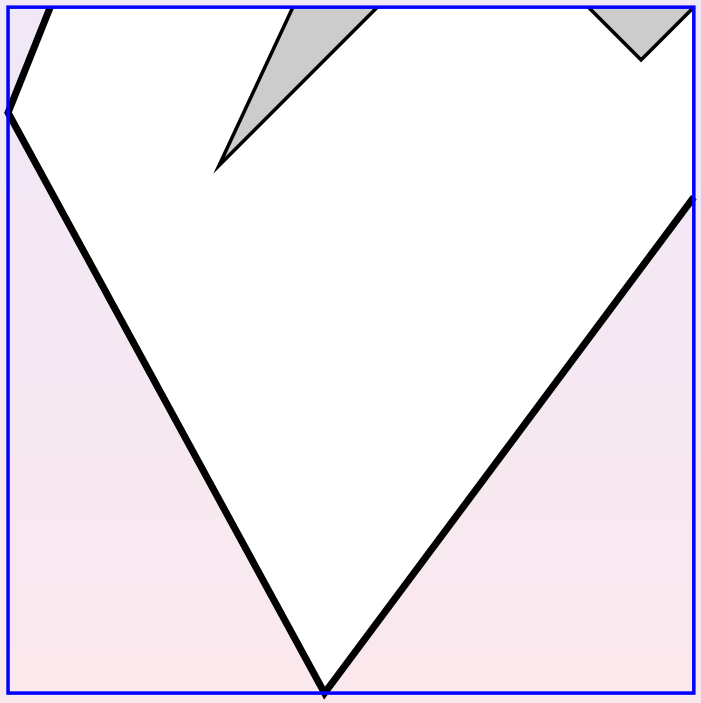
Local complexity: $O(P_c + A_c/F + k_c \cdot F)$
(P_c = perimeter; D_c = diameter; k_c = # obstacles)

Cell exploration

Basic idea: use the unlimited vision naive algorithm

Naive algorithm

- Reach the boundary of cell c
- Follow/explore the boundary
- **While** unvisited obstacles
  - Go to an unvisited obstacle
  - Go around it



Local complexity: $O(P_c + A_c/F + k_c \cdot F)$
(P_c = perimeter; D_c = diameter; k_c = # obstacles)

Upper bound (2)

Complexity of LET(F): $O(P + A/F + k \cdot F)$
Goal (lower bound): $\Omega(P + A + \sqrt{A \cdot k})$

Solution's intuition: choose $F = \min\{1, \sqrt{\frac{A}{k}}\}$

If only A is known (case when k known is similar):

Algorithm LET_A

- Set $F = 1$
- Apply LET(F) **until** the error is "too" large
- Decrease F appropriately
- Restart the algorithm with the new F

Upper bound (2)

Complexity of LET(F): $O(P + A/F + k \cdot F)$
Goal (lower bound): $\Omega(P + A + \sqrt{A \cdot k})$

Solution's intuition: choose $F = \min\{1, \sqrt{\frac{A}{k}}\}$

If only A is known (case when k known is similar):

Algorithm LET_A

- Set $F = 1$
- Apply LET(F) **until** the error is "too" large
- Decrease F appropriately
- Restart the algorithm with the new F

Upper bound (2)

Complexity of LET(F): $O(P + A/F + k \cdot F)$
Goal (lower bound): $\Omega(P + A + \sqrt{A \cdot k})$

Solution's intuition: choose $F = \min\{1, \sqrt{\frac{A}{k}}\}$

If only A is known (case when k known is similar):

Algorithm LET$_A$

- Set $F = 1$
- Apply LET(F) **until** the error is "too" large
- Decrease F appropriately
- Restart the algorithm with the new F

Complexity analysis of LET_A (A known)

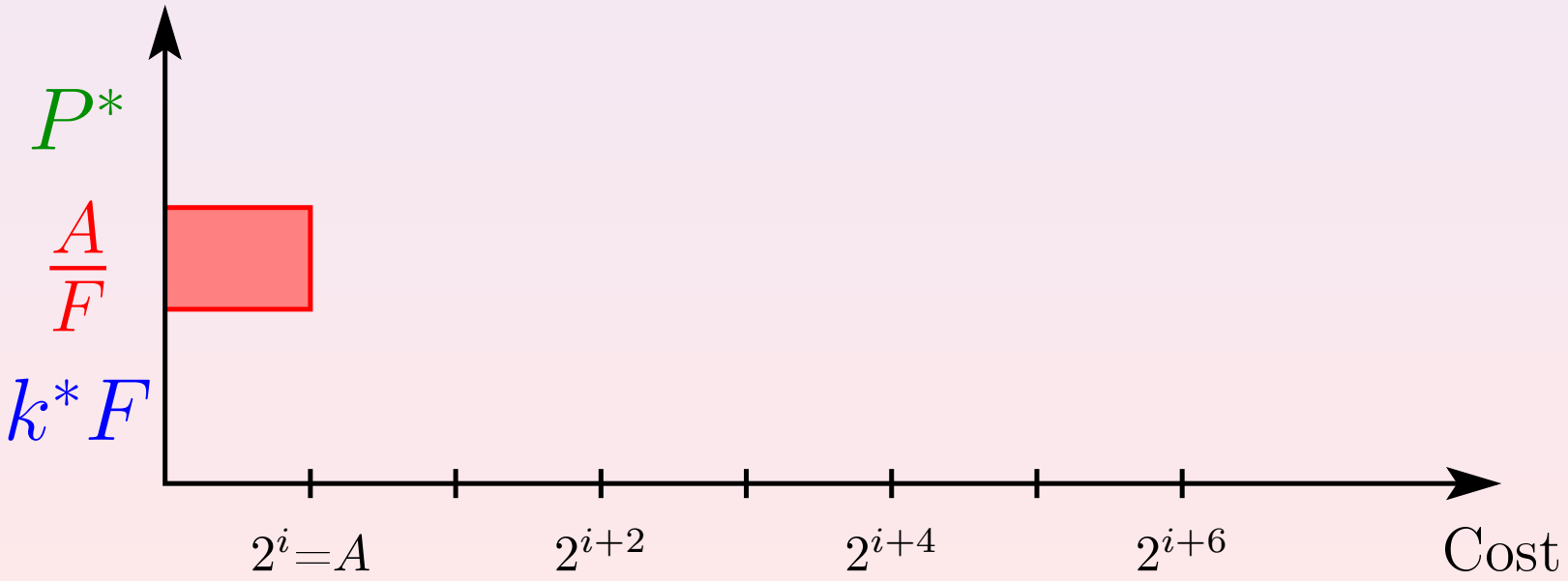
P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$

Complexity analysis of LET_A (A known)

P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

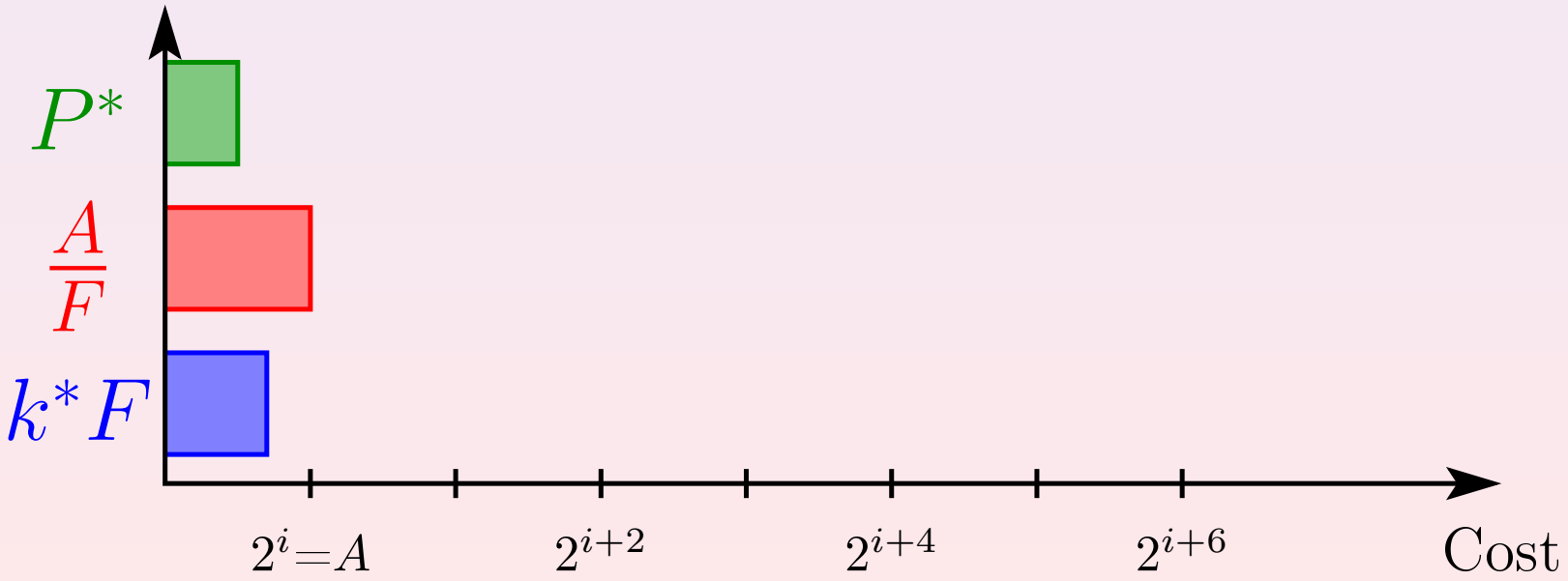
- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$



Complexity analysis of LET_A (A known)

P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

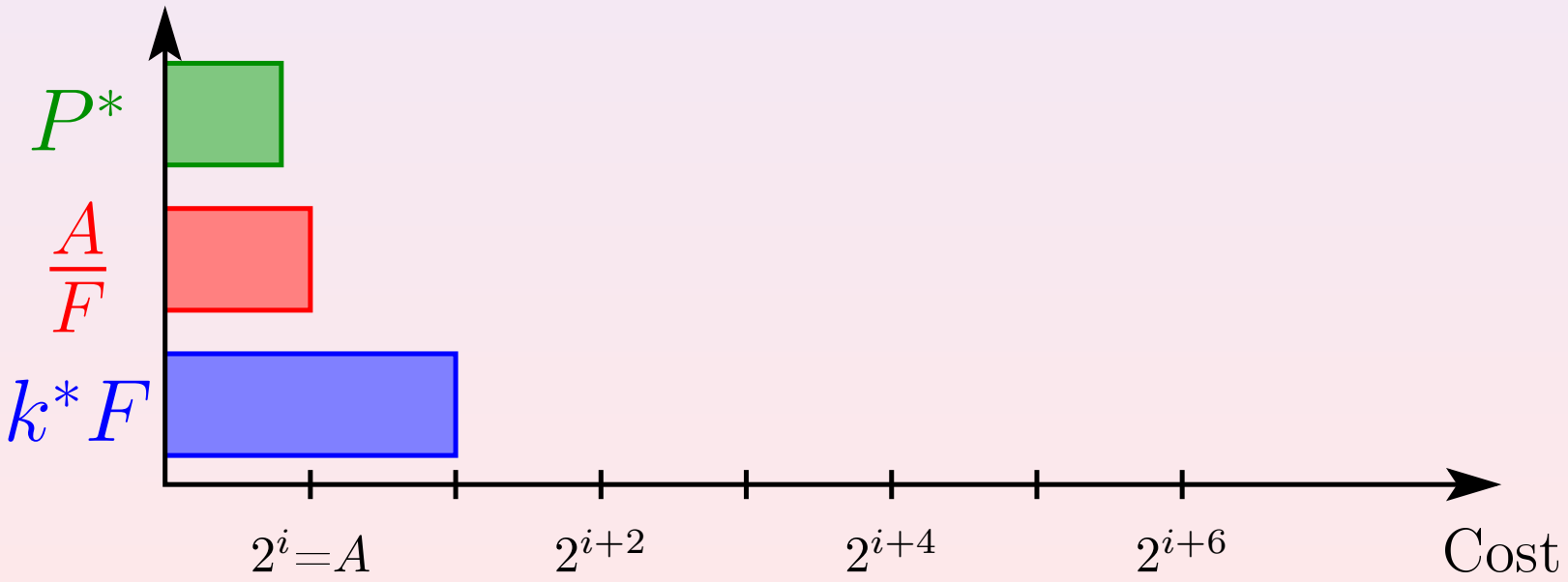
- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$



Complexity analysis of LET_A (A known)

P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

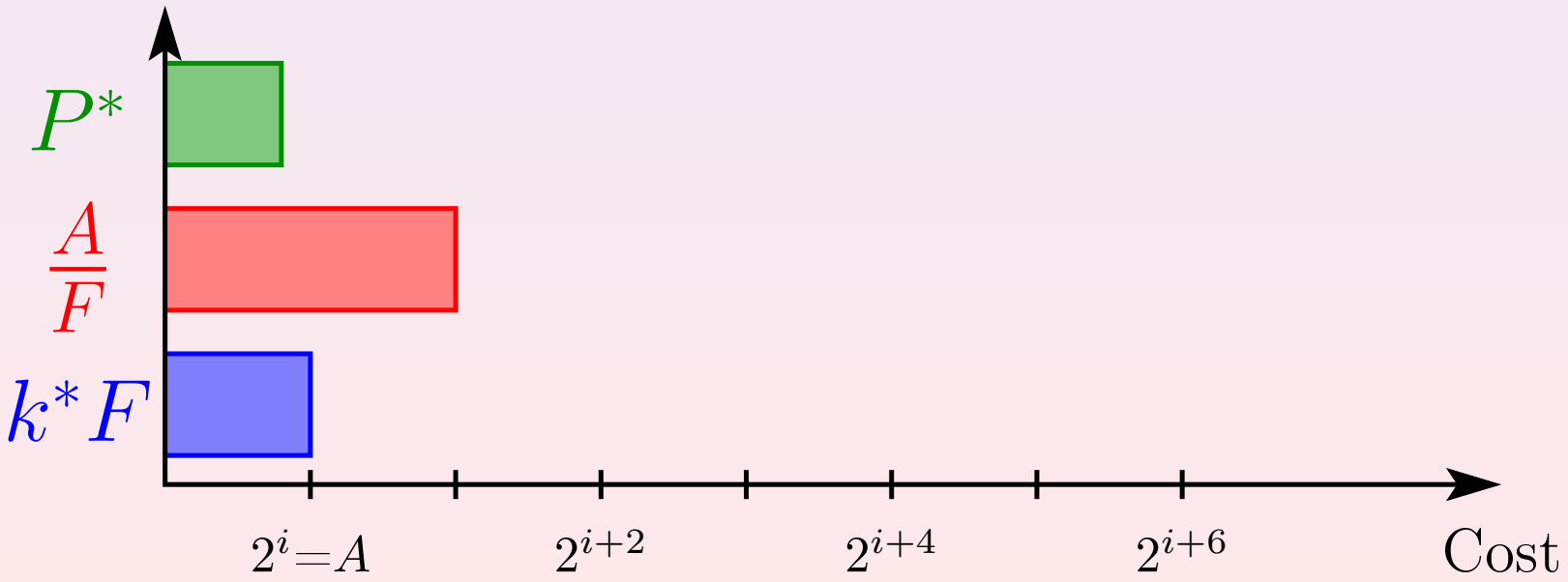
- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$



Complexity analysis of LET_A (A known)

P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

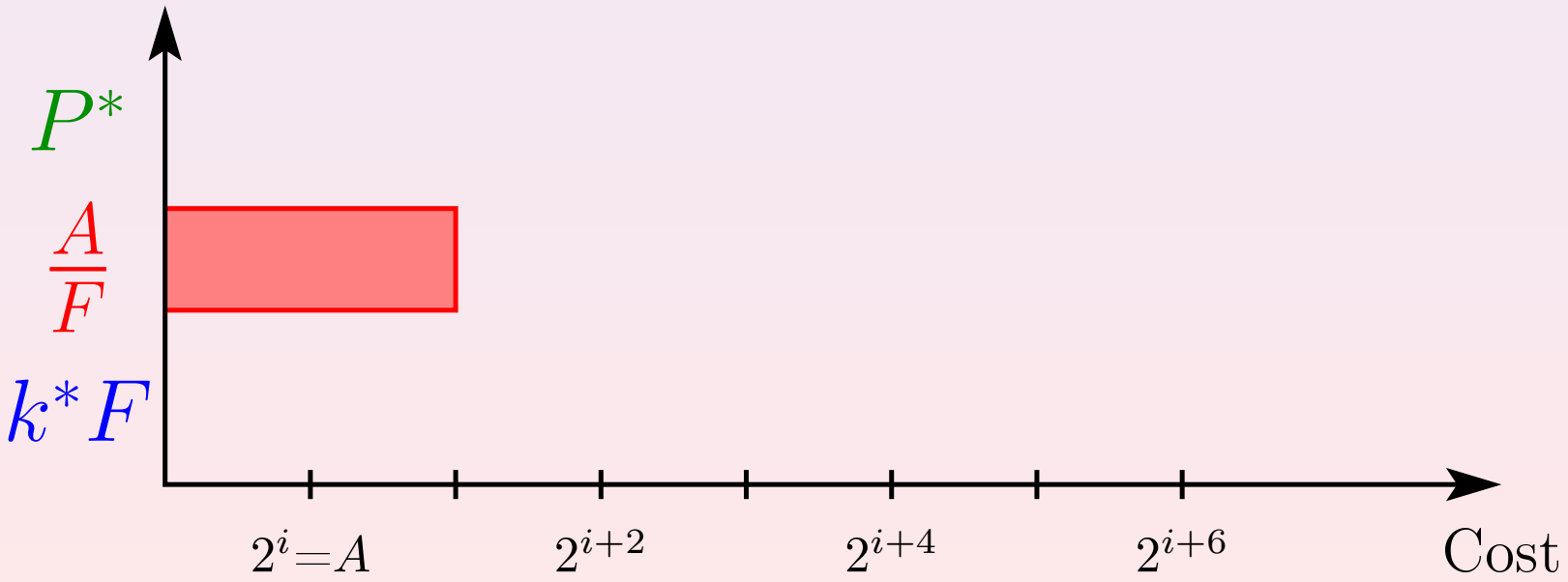
- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$



Complexity analysis of LET_A (A known)

P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

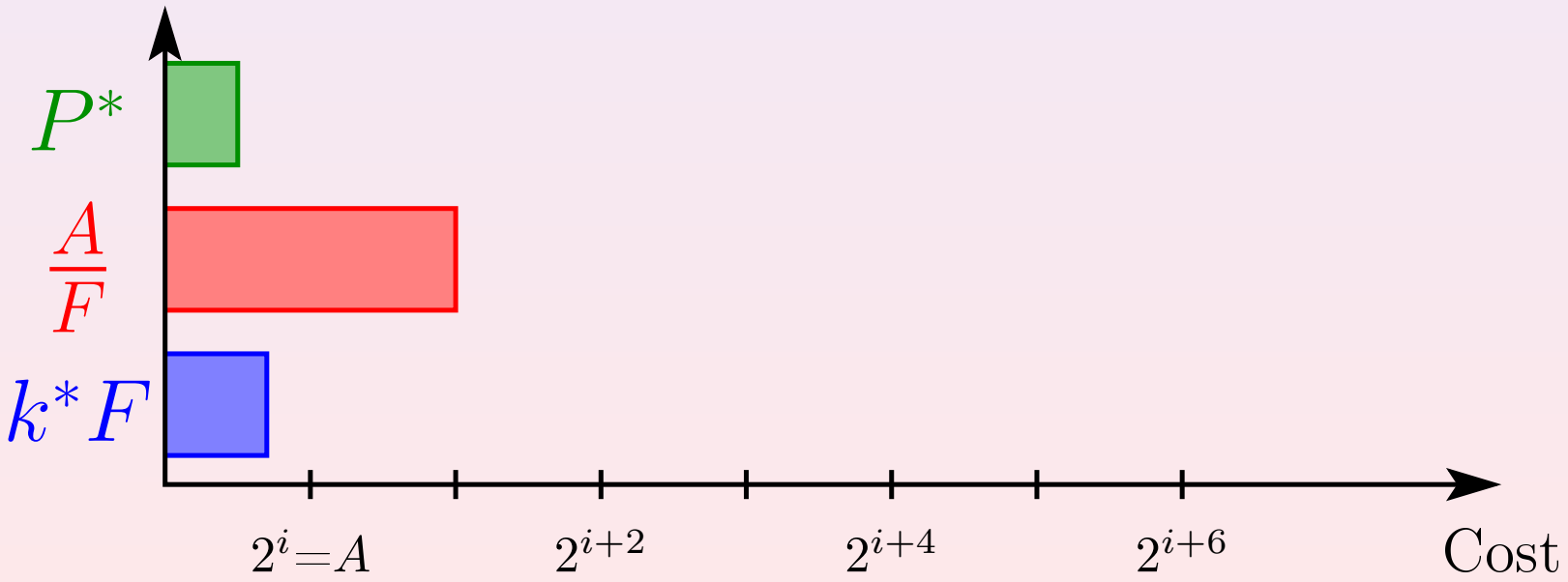
- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$



Complexity analysis of LET_A (A known)

P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

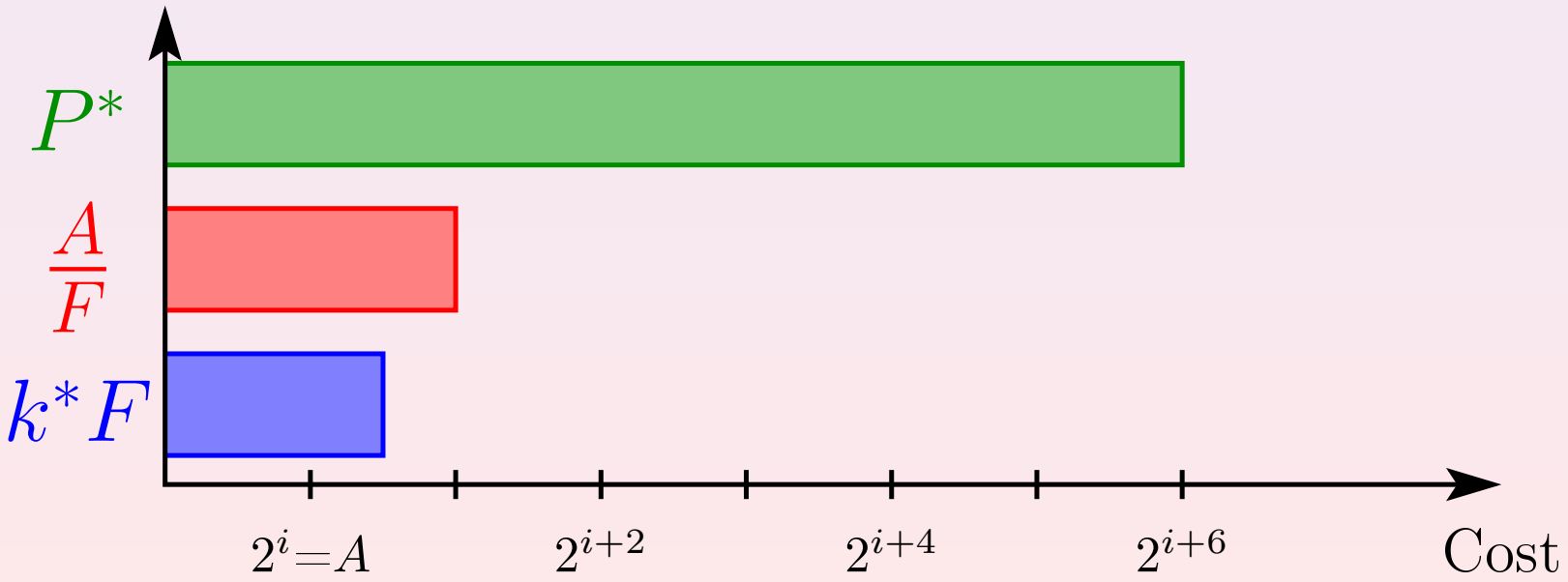
- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$



Complexity analysis of LET_A (A known)

P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

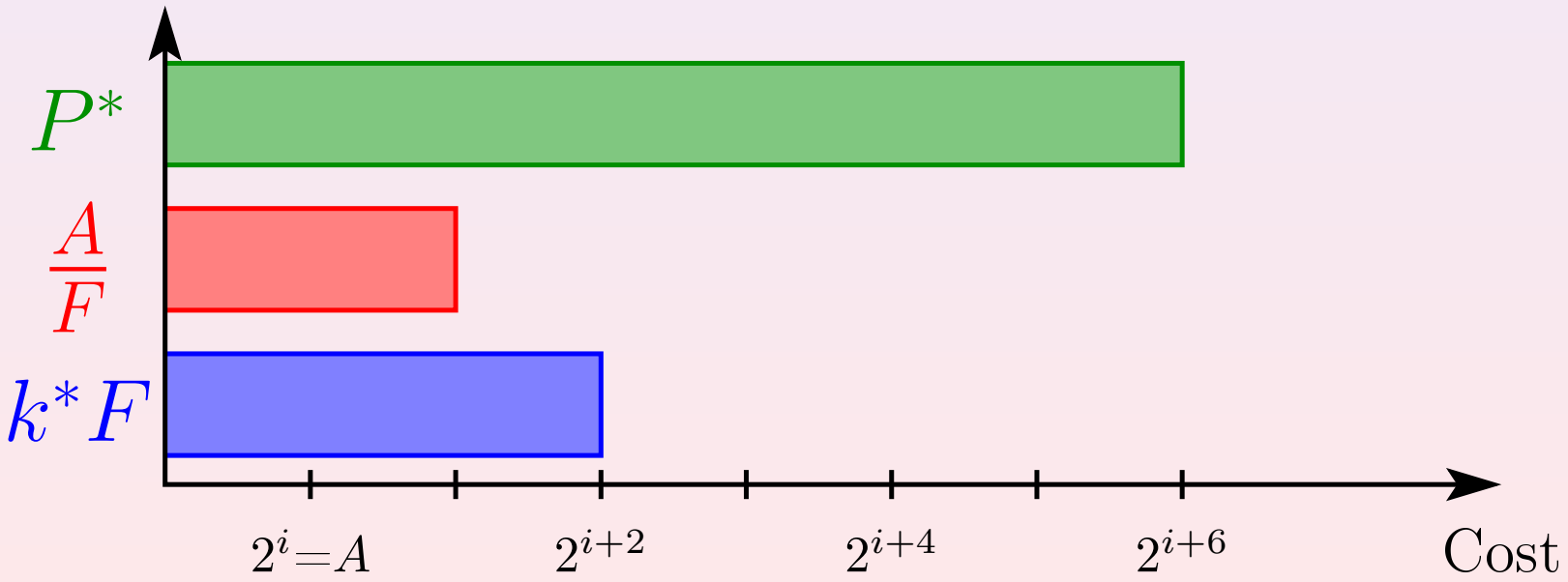
- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$



Complexity analysis of LET_A (A known)

P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

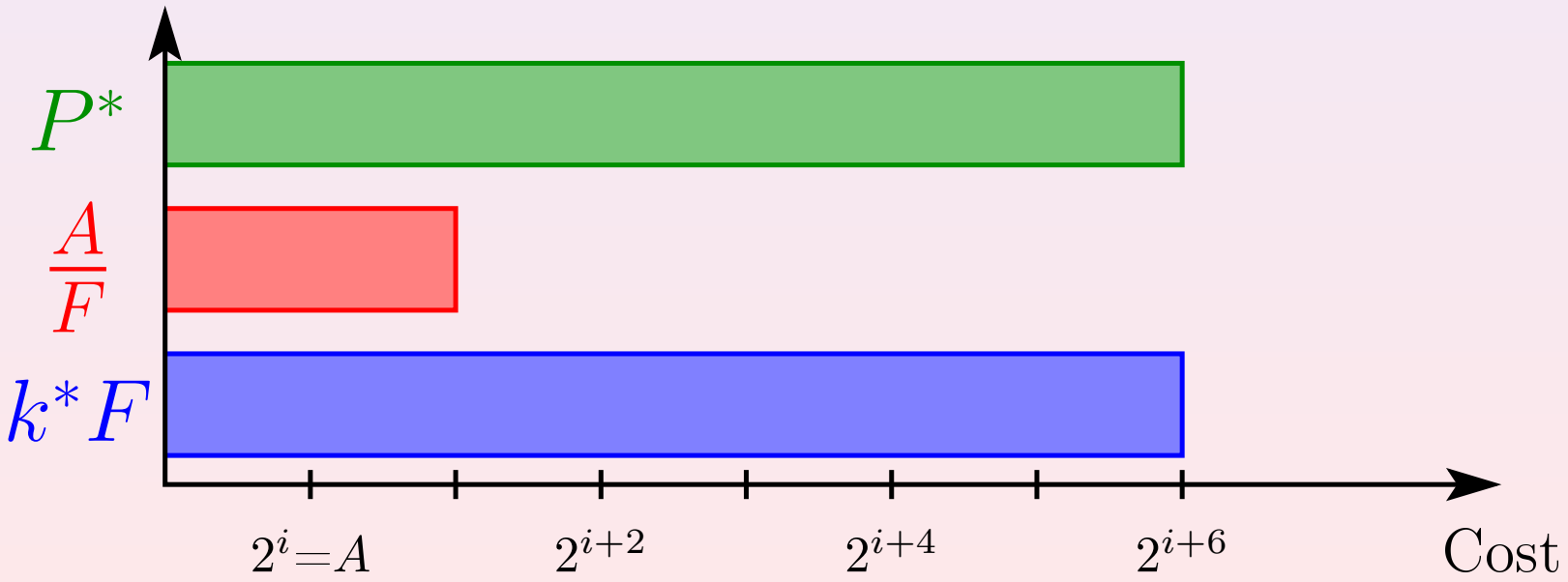
- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$



Complexity analysis of LET_A (A known)

P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

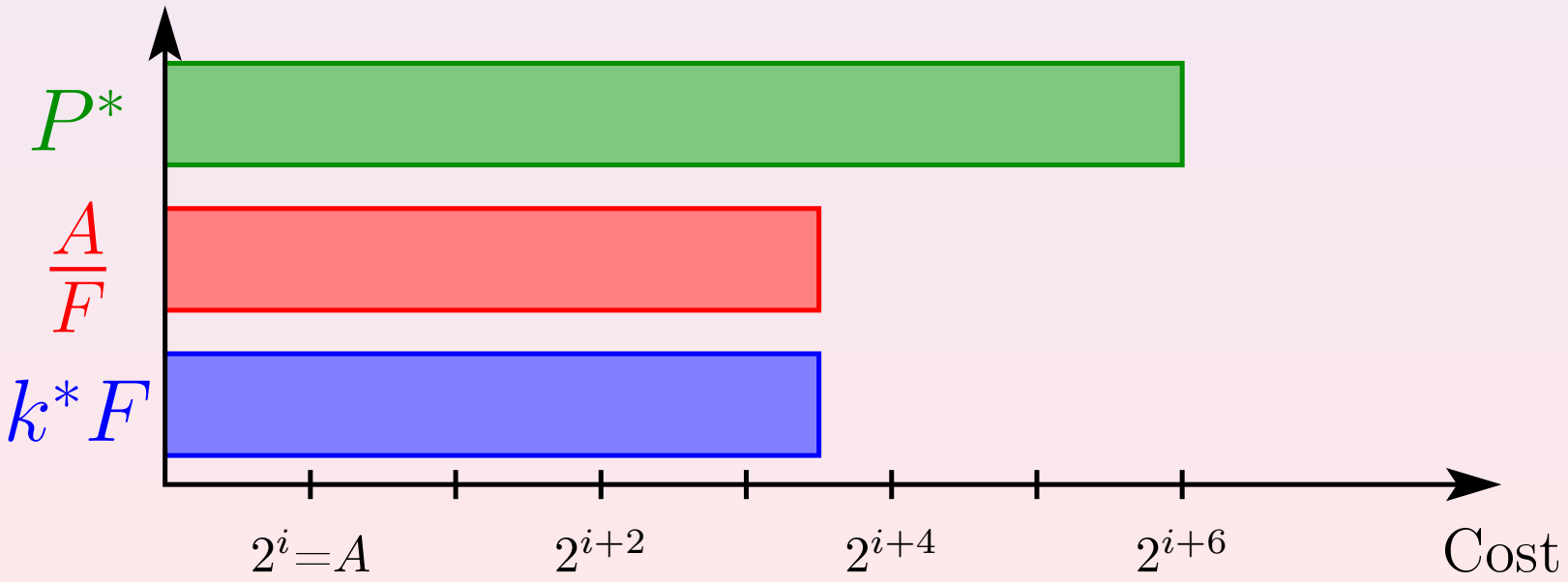
- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$



Complexity analysis of LET_A (A known)

P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

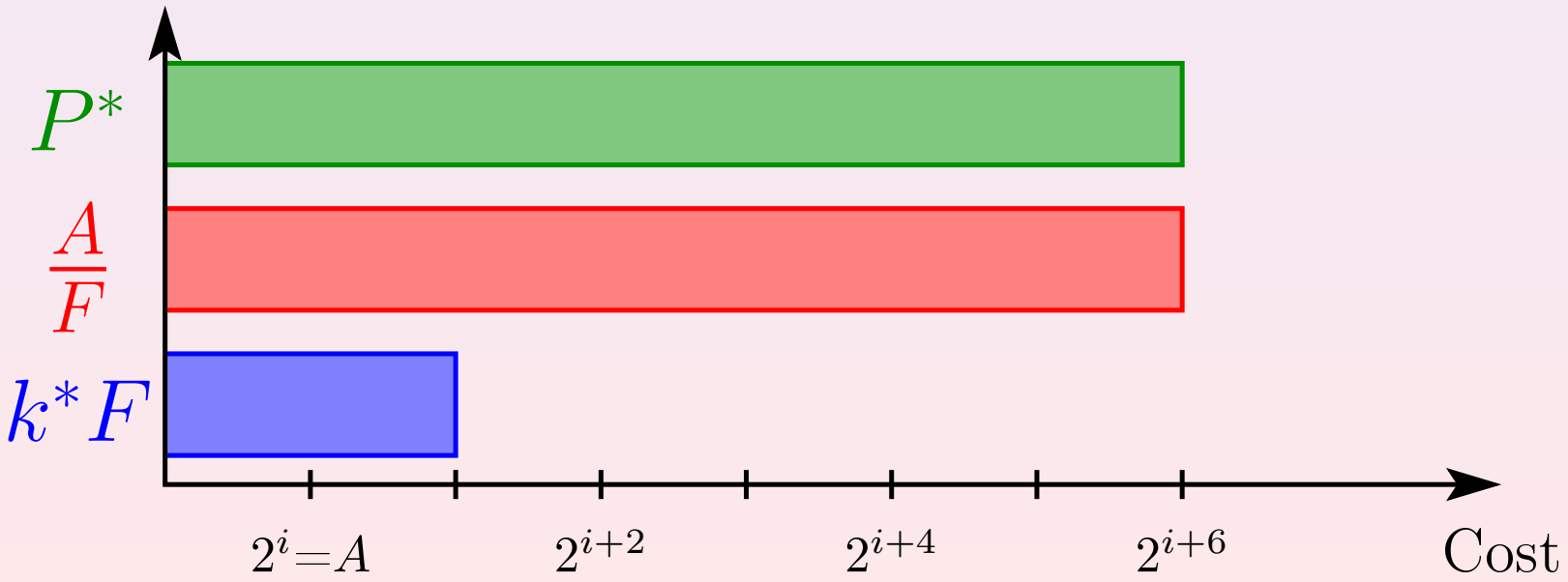
- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$



Complexity analysis of LET_A (A known)

P^*, k^*: current discovered values of P and k
P_j, k_j: values of P^*, k^* at the end of Phase j

- $F_1 = 1$
- Stopping condition: $\{k^* F_j \geq 2\frac{A}{F_j}$ and $k^* F_j \geq P^*\}$
- F's update: $F_{j+1} = \frac{A}{k_j F_j}$



Outline

Conclusion and perspectives

(Reminder: P = perimeter; A = area; k = # obstacles)

Limited vision

- $\Omega(P + A + \sqrt{Ak})$, even if the terrain is a priori known
- $O(P + A + \sqrt{Ak})$, if k or A is known (algorithm)
- $O(\min\{\log A, \log k\}(P + A + \sqrt{Ak}))$, nothing known

Open problem

- *Worst-case complexity* without knowledge?
- *Competitive ratio* in any polygonal terrain with polygonal obstacles (limited and unlimited)?

Conclusion and perspectives

(Reminder: P = perimeter; A = area; k = # obstacles)

Limited vision

- $\Omega(P + A + \sqrt{Ak})$, even if the terrain is a priori known
- $O(P + A + \sqrt{Ak})$, if k or A is known (algorithm)
- $O(\min\{\log A, \log k\}(P + A + \sqrt{Ak}))$, nothing known

Open problem

- *Worst-case complexity* without knowledge?
- *Competitive ratio* in any polygonal terrain with polygonal obstacles (limited and unlimited)?

Conclusion and perspectives

(Reminder: P = perimeter; A = area; k = # obstacles)

Limited vision

- $\Omega(P + A + \sqrt{Ak})$, even if the terrain is a priori known
- $O(P + A + \sqrt{Ak})$, if k or A is known (algorithm)
- $O(\min\{\log A, \log k\}(P + A + \sqrt{Ak}))$, nothing known

Open problem

- *Worst-case complexity* without knowledge?
- *Competitive ratio* in any polygonal terrain with polygonal obstacles (limited and unlimited)?

Thank You
for your attention