Uniform dispersal of robots with minimum visibility range

Attila Hideg[1] and Tamás Lukovszki[2]

1 Budapest University of Technology and Economics, Hungary
Attila.Hideg@aut.bme.hu

2 Eötvös Lóránd University, Budapest, Hungary
lukovszki@inf.elte.hu

Filling

- n robots with restricted capabilities
  - >Limited sensing range
  - >No communcation
  - >O(1) bits memory
- 2D plane setting
  - >Orthogonal area (2D grid)
  - >Unknown, connected

https://ssr.seas.harvard.edu/

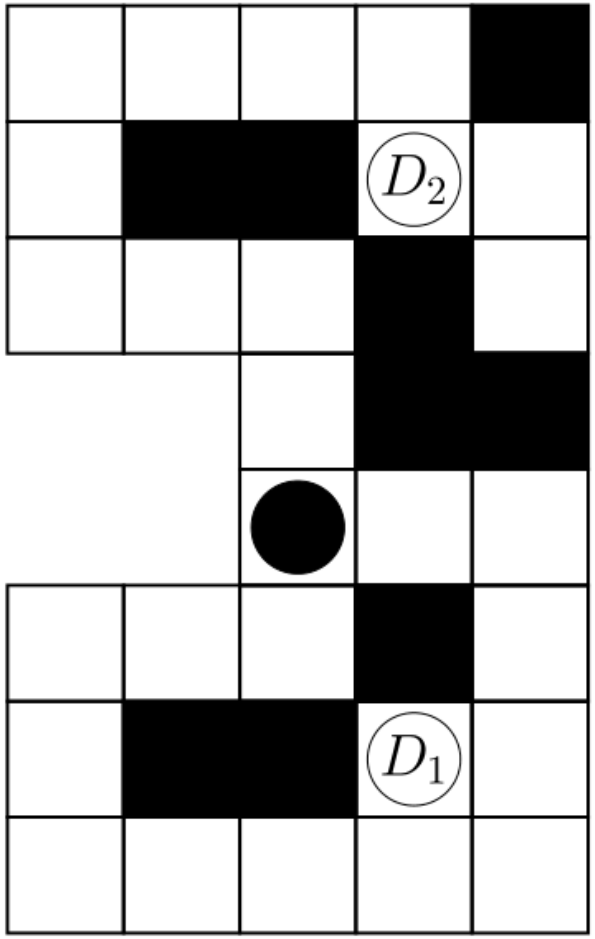
- Robots enter at the Door (or multiple Doors)
  - >When a Door becomes empty, a robot is placed immediately
- They have to cover the area

Anonymous Restricted Robots

- Identical and anonymous
- Silent
- 1 hop visibility
- Common notion of
- up-down and left-right
- Orthogonal area
  >Decomposed into square-cells
- Robots can move to neighboring cells
  >i.e. cells sharing a side

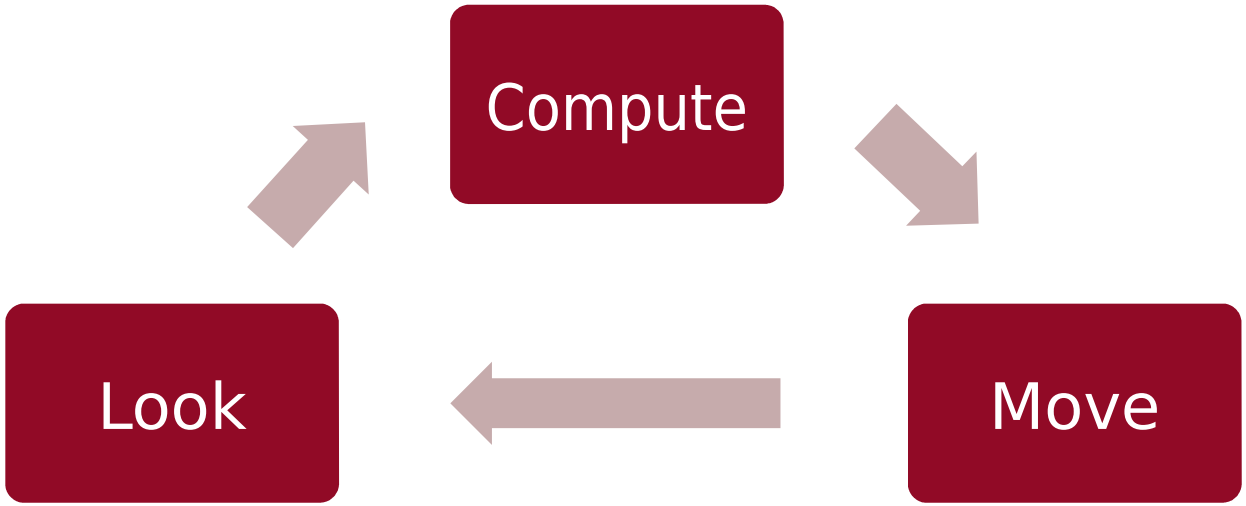
Anonymous Restricted Robots

- Identical and anonymous
- Silent
- 1 hop visibility
- Common notion of
- up-down and left-right
- Orthogonal area
  >Decomposed into square-cells
- Robots can move to neighboring cells
  >i.e. cells sharing a side



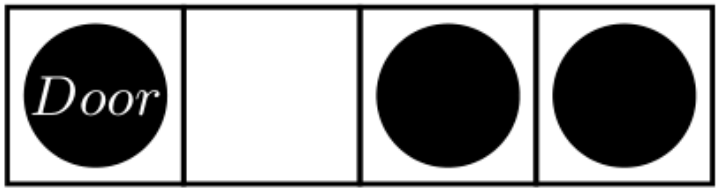
Synchronous Look-Compute-Move (LCM)

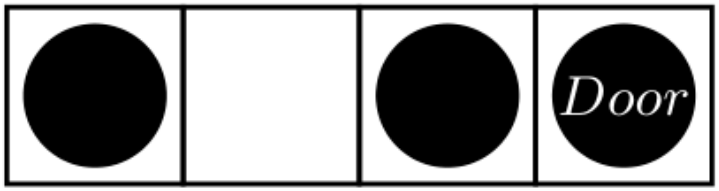
- Look: take a snapshot
- Compute: calculate the destinatin
- Move: move to the destination



Lower bounds

- Visibility: 1 hop
- W(n) running time
- Memory: O(1) bits [Barrameda et al. 2008]





State of the art

- Visibility range: # hops
- Communcation range: # hops
- Memory: # bits

Method	Doors	Visibility	Comm.	Memory	Area
DFLF [Hsiang et al. 2004]]	Single	2	2	2	Arbitrary
TALK [Barrameda et al 2013]	Single	2	2	4	Orthogonal
MUTE [Barrameda et al 2013]	Single	6	0	9	Orthogonal
MULTIPLE [Barrameda et al 2008]	Multiple	3	0	4	Orthogonal
Single Door (new)	**Single**	**1**	**0**	**13**	**Orthogonal**
Multiple Door (new)	**Multiple**	**1**	**0**	**13**	**Orthogonal**

Our Contribution (1)

- Single Door
- Requirements
  >1 hop visibility
  >No communcation
  >13 bits memory
  >Common top-down, left-right directions
- Running time
  >O(n) rounds: 12n LCM cycles

Our Contribution (2)

- Multiple Door
- Requirements
  >1 hop visibility
  >No communcation
  >13 bits memory
  >Common top-down, left-right directions
- Running time
  >O(n) rounds: 12n LCM cycles

Method

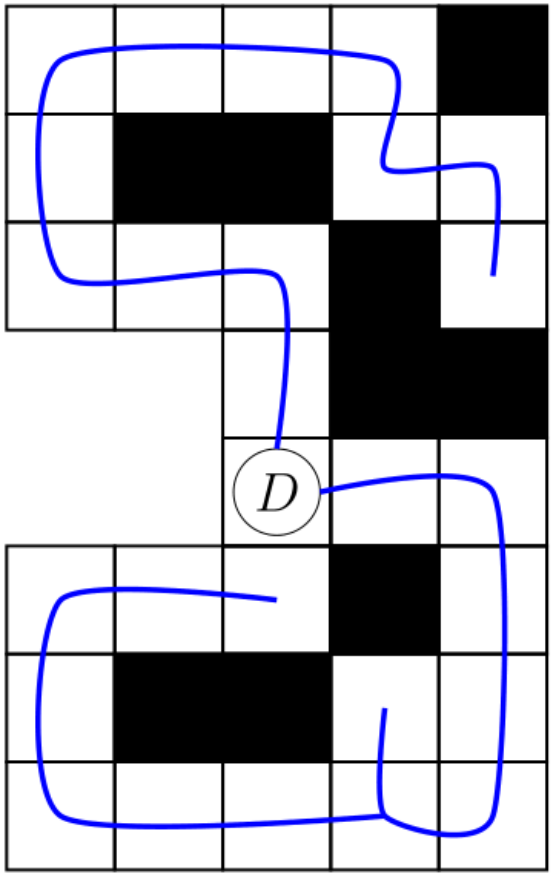
- Mimimcs a DFS traversal of the area
- Main tasks to solve:
  >Prevent collisions
  >Fill the whole area

- Main ideas:
  >Timing of movements
  >Follow the Leader method

Method – Filling the area

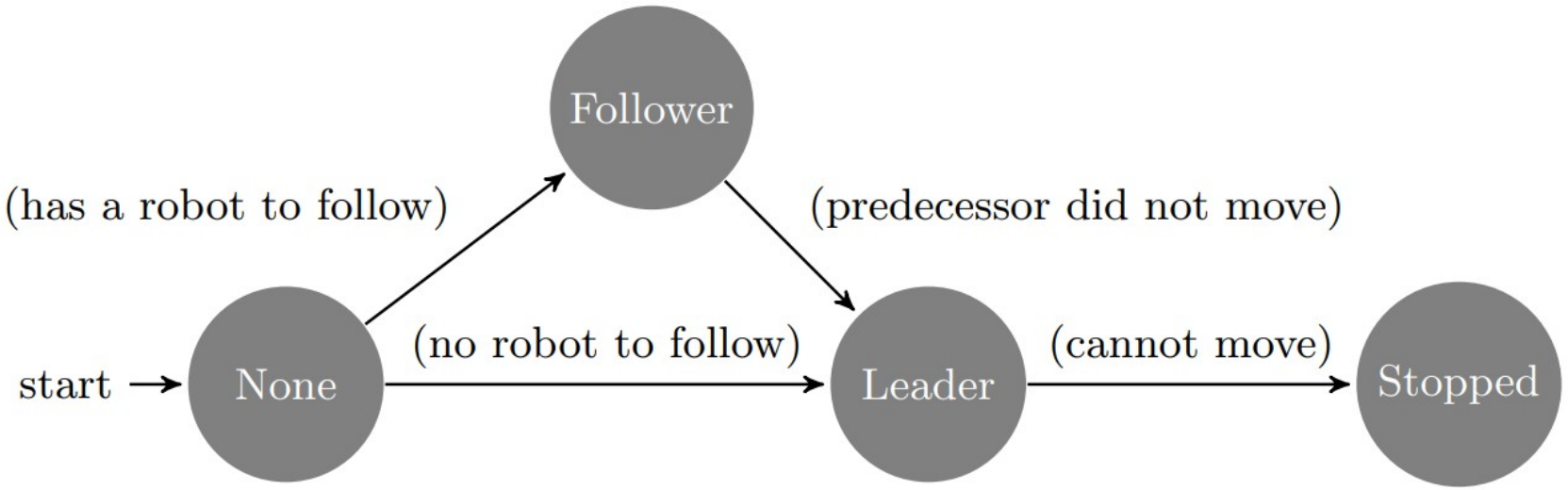
- Mimimcs a DFS traversal of the area

- DFS tree
  >Unkown for the robots
  >Traversed by the robots
  >Branches are filled



Method – States

- Leader-Follower method
- Four possible states



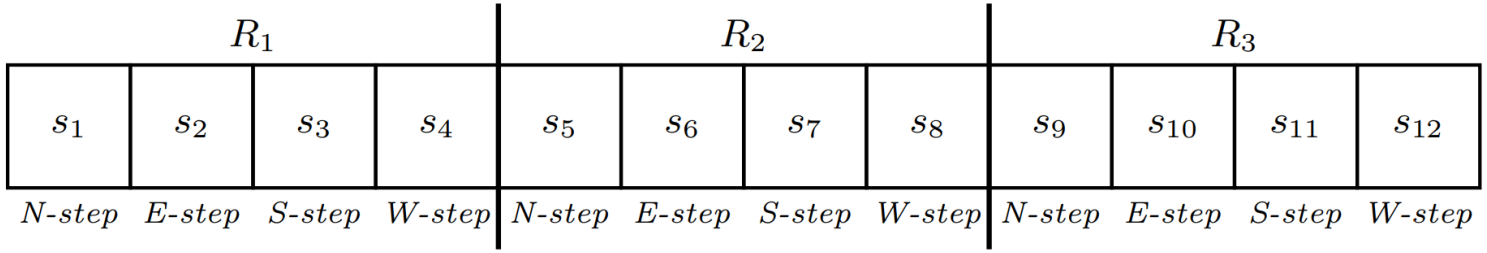
Method – Invariants

- There is at most one Leader
- When the Leader stucks,
  >the Leadership is transfered

- Followers only follow their predecessor
  >Predecessor is either in a neighboring cell or
  >It moved away, then in the next step the follower moves to its prevIous position

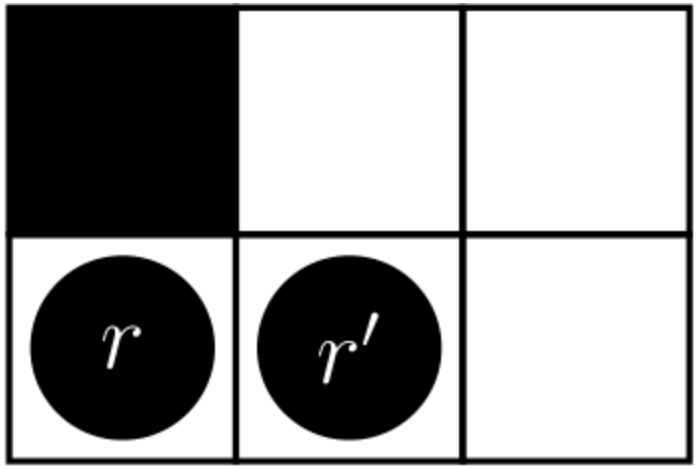
Method – Preventing Collisions

- Timing of the movements
- Step
  - >Four possible directions (North, East, South, West)
- Round
  - >4 step long time window
  - >Odd and Even rounds for the robots
  - >Odd: observing, Even: moving



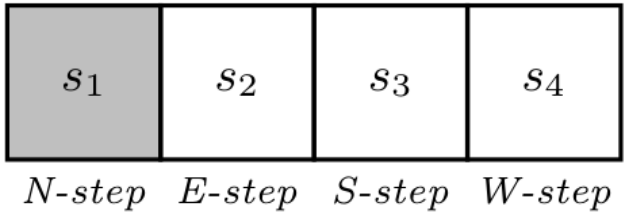
Method – Preventing Collisions

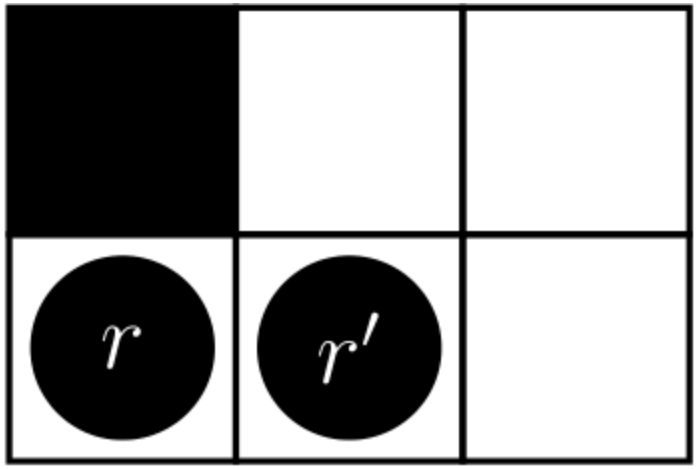
- Example
  >North step
  >r’ moves



Method – Preventing Collisions

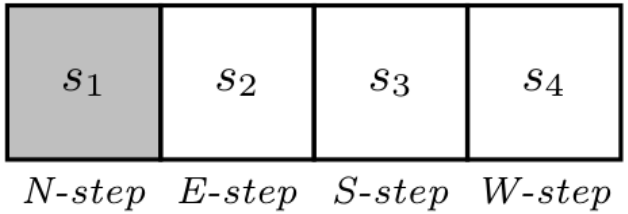
Look phase:

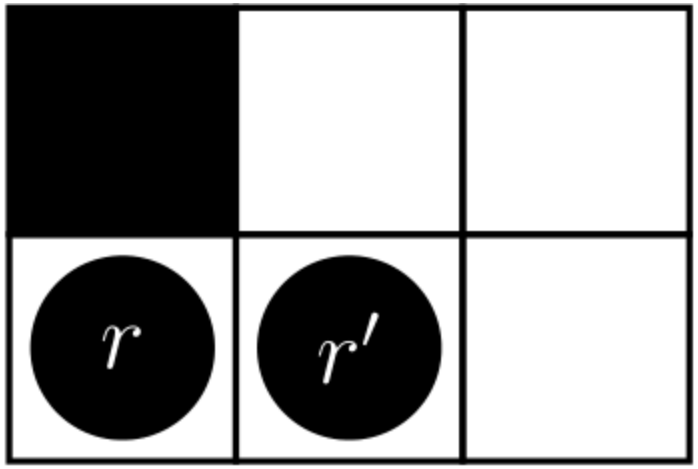




Method – Preventing Collisions

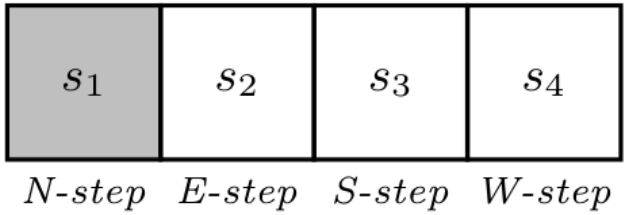
Compute phase:

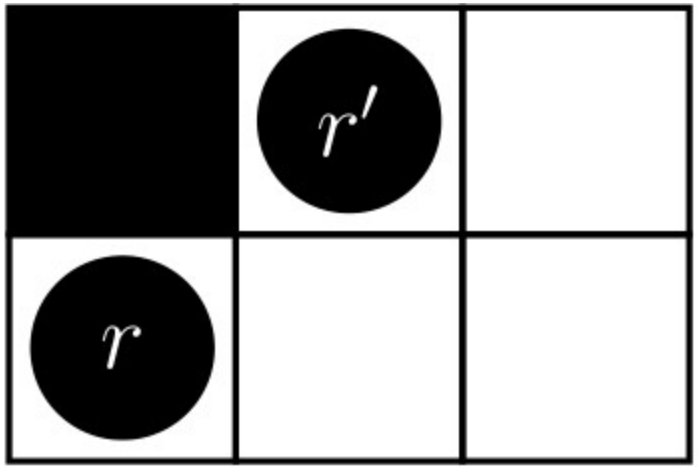




Method – Preventing Collisions

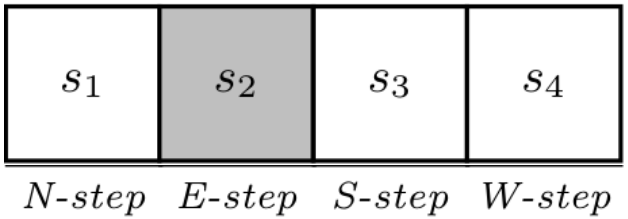
Move phase:

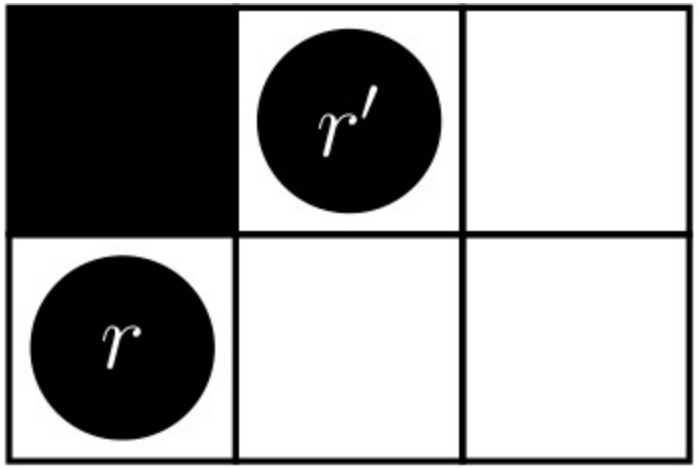




Method – Preventing Collisions

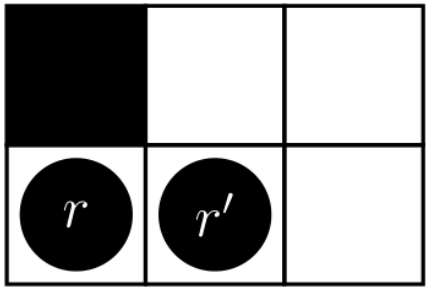
Look phase:

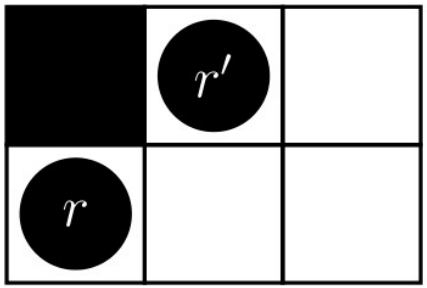




State: Follower

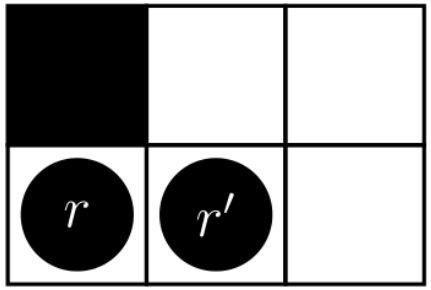
- Task: Follow its predecessor
  >One hop visibility

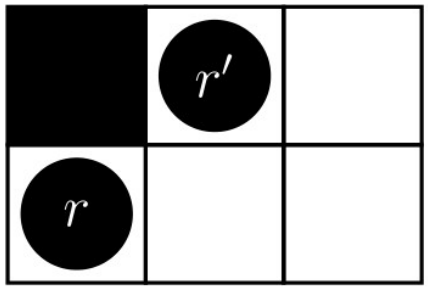




State: Follower

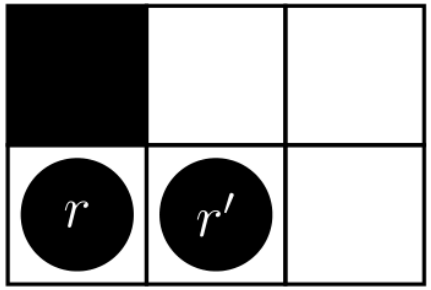
- Task: Follow its predecessor
  >One hop visibility
- Odd rounds
  >Observes the predecessor
    - and empty neighbors
    - Stores which direction the predecessor moves
  >(known from the timing!)

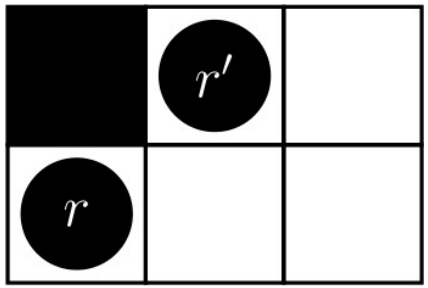




State: Follower

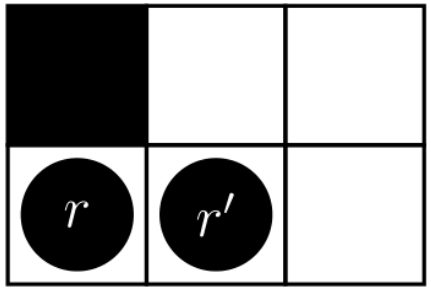
- Task: Follow its predecessor
  >One hop visibility
- Odd rounds
  >Observes the predecessor
    - and empty neighbors
    - Stores which direction the predecessor moves
  >(known from the timing!)
- Even rounds
  >Moves to the previous position of the predecessor

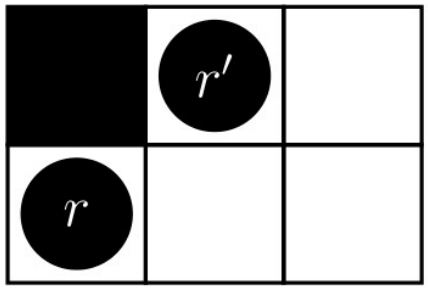




State: Follower

- Task: Follow its predecessor
  >One hop visibility
- Odd rounds
  >Observes the predecessor
    - and empty neighbors
    - Stores which direction the predecessor moves
  >(known from the timing!)
- Even rounds
  >Moves to the previous position of the predecessor
- If predecessor did not move
  >Switches to Leader state
  >Leadership transferred





State: Leader

- Task
  >Has to move to free cells

- Odd rounds
  >Takes a snapshot, and stores occupied cells
- Even round
  >Moves to the cell corrseponding to the direction of the step if it is unoccupied

- Switch to Stopped if no free cells around

State: None

- Robot placed at the Door
  - >After predecessor moved from it
  - >Initial state is None
  - >Knows which step is the currently performed (N,E,S,W)
- Predecessor moved in even round
  - >None-state robot must have its odd round in the next round
  - >Idle in the current round and
  - >becomes a Follower in the next round

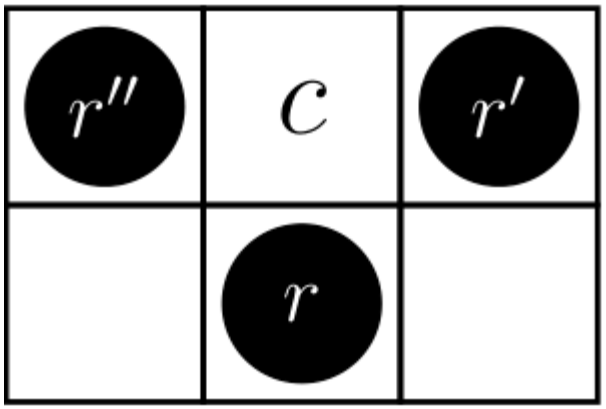
Result

Theorem 1: The algorithm fills a connected orthogonal area with a single door

- In O(n) rounds
- Requirements
  >Visibility range of 1 hop
  >O(1) bits of persistent memory
  >Common notion of North, South, East, West

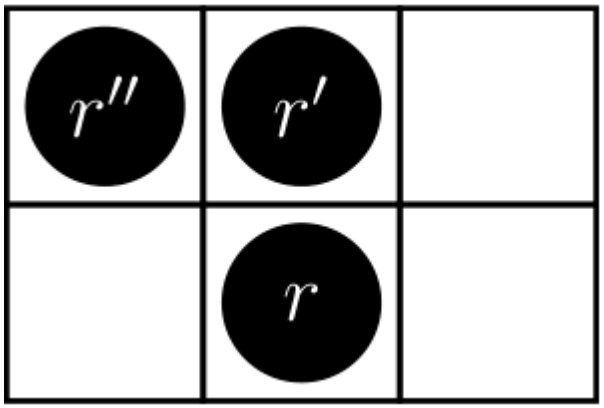
Analysis

- The Leader only moves to free cells
  >Takes a snapshot in odd rounds
  >Knows which ones are free in even rounds



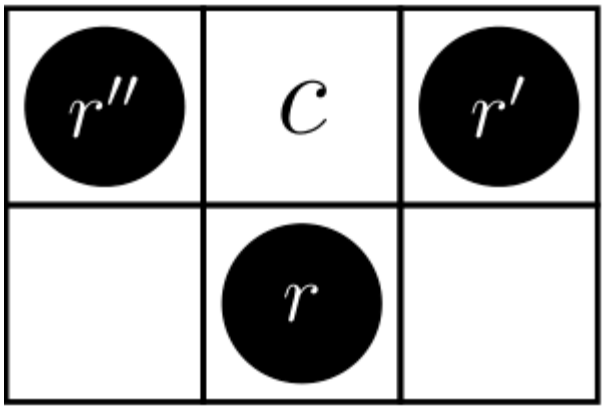
Analysis

- The Leader only moves to free cells
  >Takes a snapshot in odd rounds
  >Knows which ones are free in even rounds



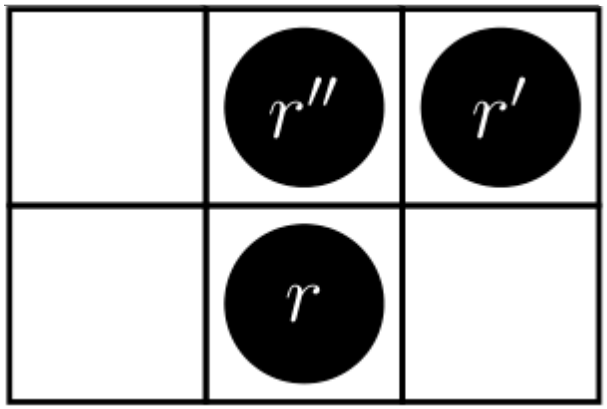
Analysis

- The Leader only moves to free cells
  >Takes a snapshot in odd rounds
  >Knows which ones are free in even rounds



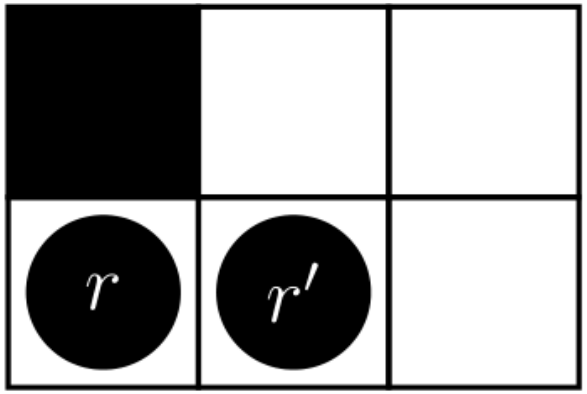
Analysis

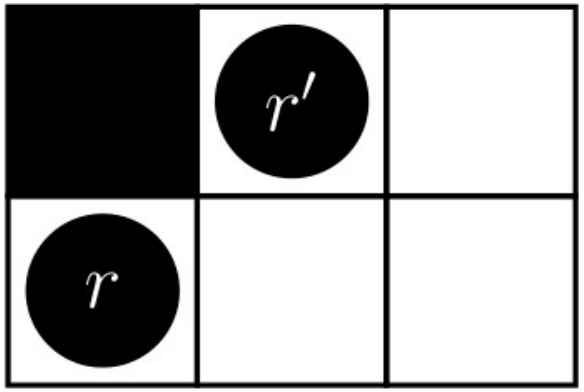
- The Leader only moves to free cells
  >Takes a snapshot in odd rounds
  >Knows which ones are free in even rounds



Analysis

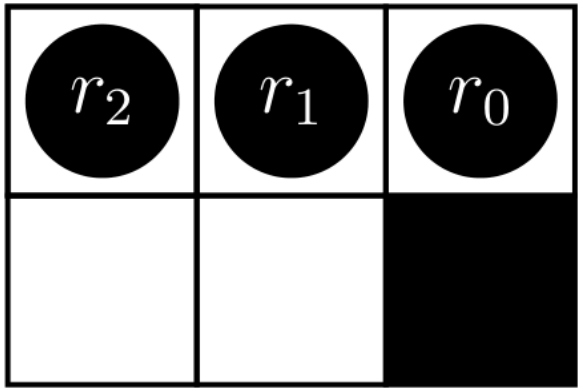
- Each Follower knows where its predecessor is
  - >Odd round:
    - knows which neighbor
    - observes its movement
  - >Even round: follows





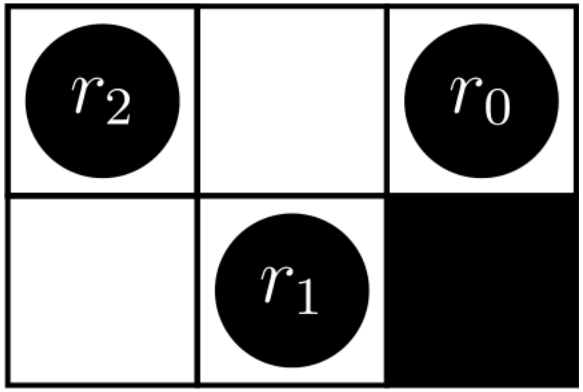
Analysis

- At most one Leader is present in the area
  >Leadership transferred



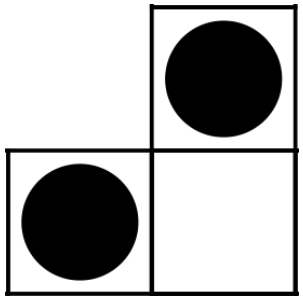
Analysis

- At most one Leader is present in the area
  >Leadership transferred



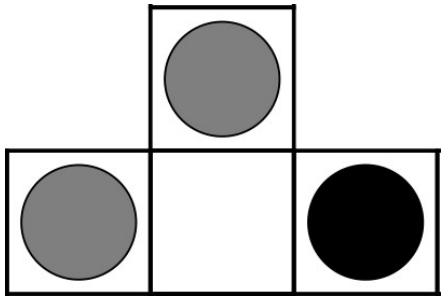
Analysis

- No collisions can occur during the dispersion



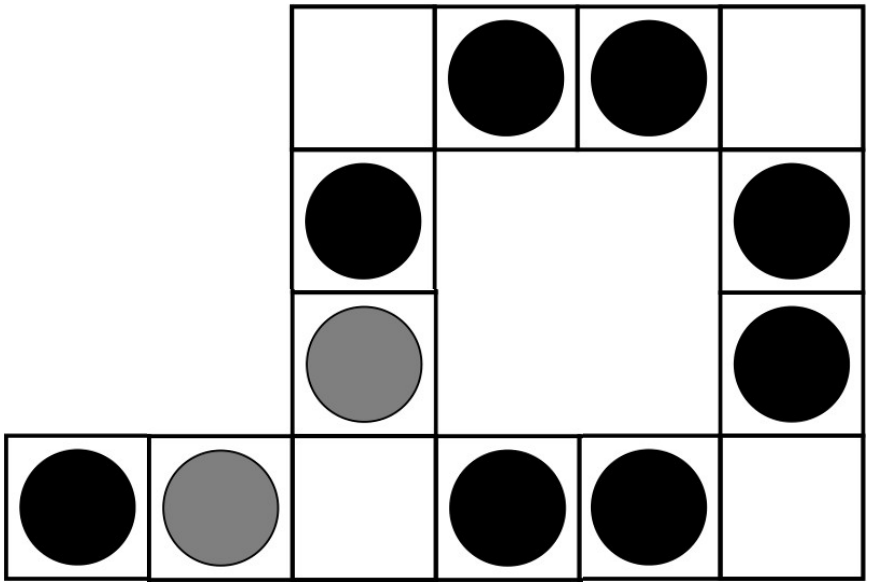
Analysis

- No collisions can occur during the dispersion
  >Follower: follows its unique predecessor



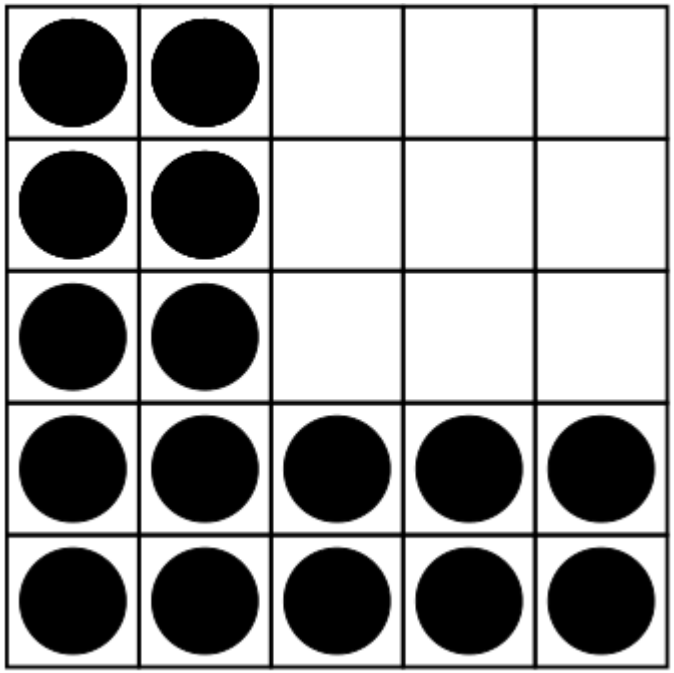
Analysis

- No collisions can occur during the dispersion
  >Follower: follows its unique predecessor
  >Leader: free cells



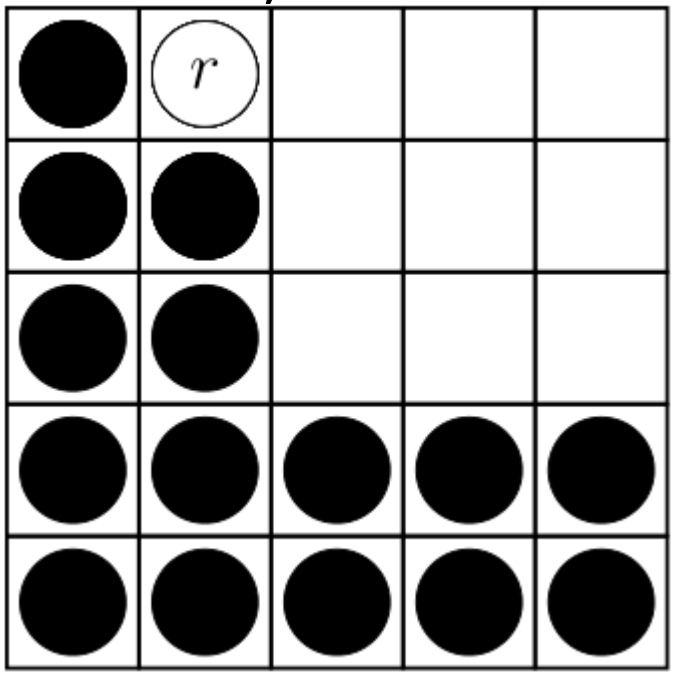
Analysis

- The proposed method fills the area
- For contradiction: Assume robots terminated and
  >the area is not filled



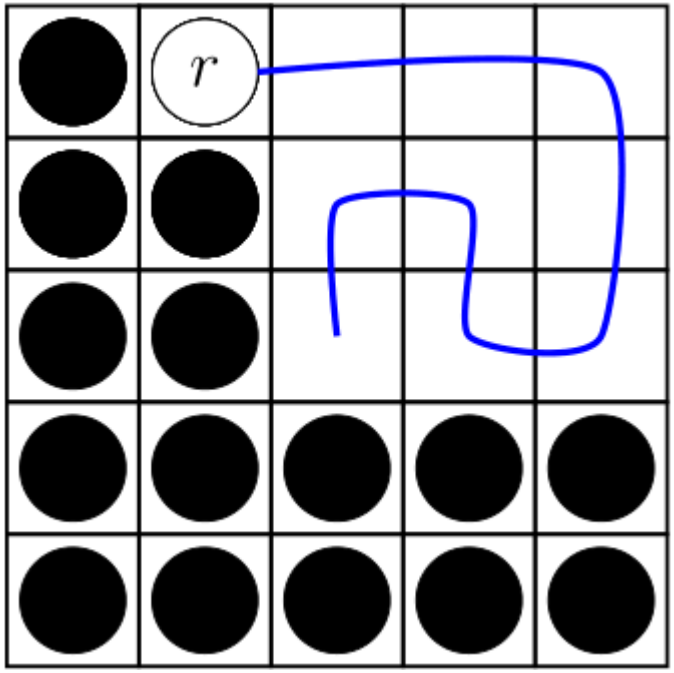
Analysis

- The proposed method fills the area
  - >*r* : first robot terminating which has an empty neighbor
  - >Before *r* terminated, *r* was a Leader



Analysis

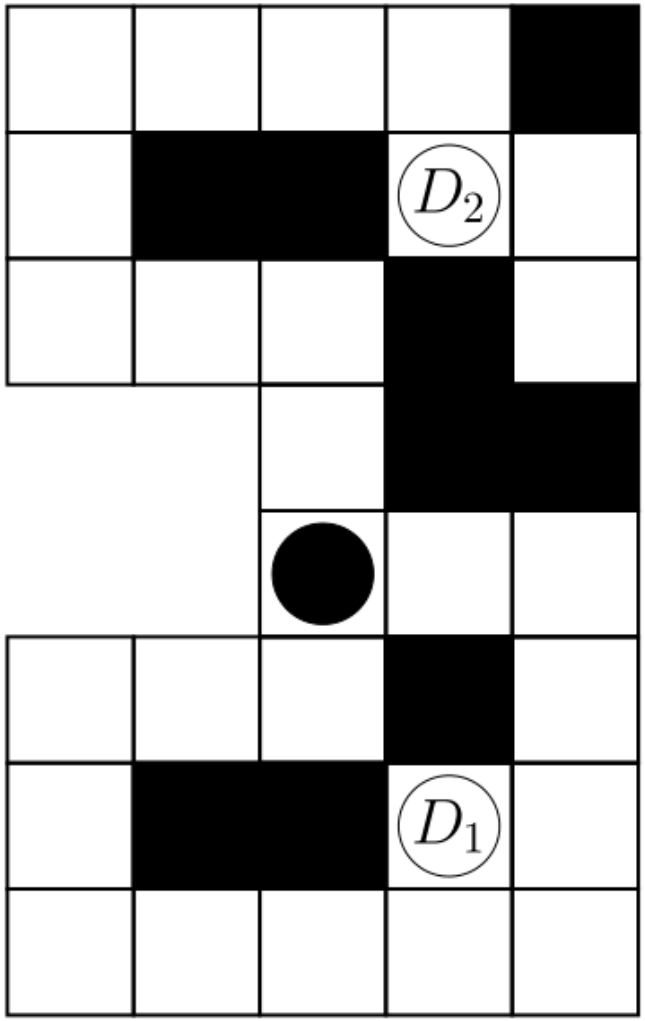
- The proposed method fills the area
- Contradiction: robots cannot terminate if unoccupied cells are present



k-Door Filling

Multiple Door or
k-Door filling

- Assume each Door has
- enough robots
- A speed-up by *k* is possible



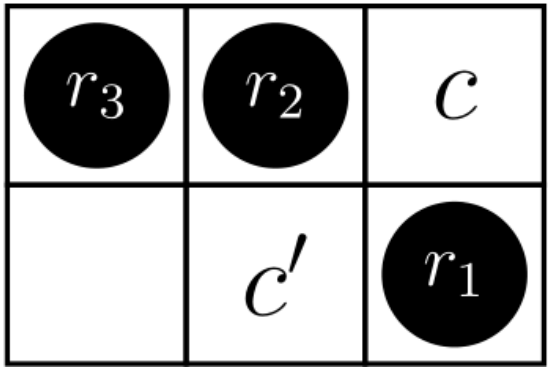
k-Door Filling

Theorem 2: The algorithm fills a connected orthogonal area with multiple Doors

- In O(n) rounds
- Requirements
  - >Visibility range of 1 hop
  - >O(1) bits of persistent memory
  - >Common notion of North, South, East, West

Analysis

- A Leader cannot collide with another Leader



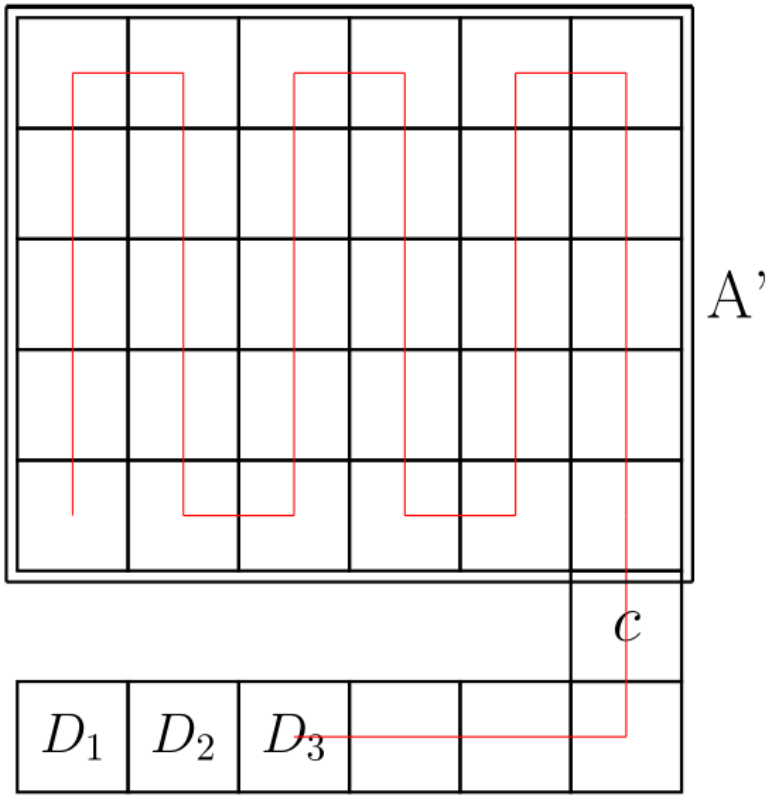
Analysis

- A Leader cannot collide with a Follower
  >Leaders can determine free cells

- Paths of different Leaders cannot cross each other
  >Leader moves to free cells
  >Leaders cannot collide

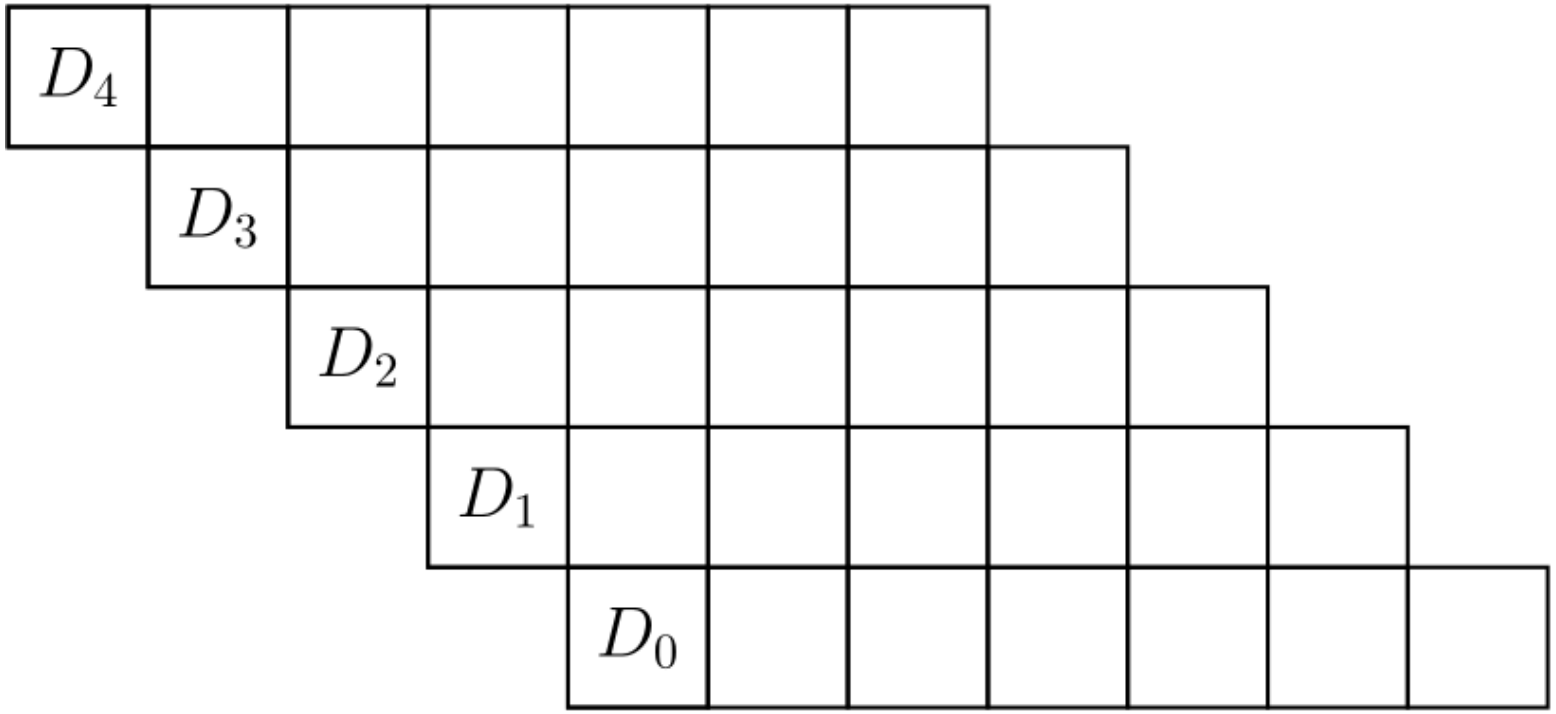
Analysis

- k speed-up is possible
- Worst-case optimal
  - >c is a bootleneck
  - >Only robots from D_3 are
  - >filling the area



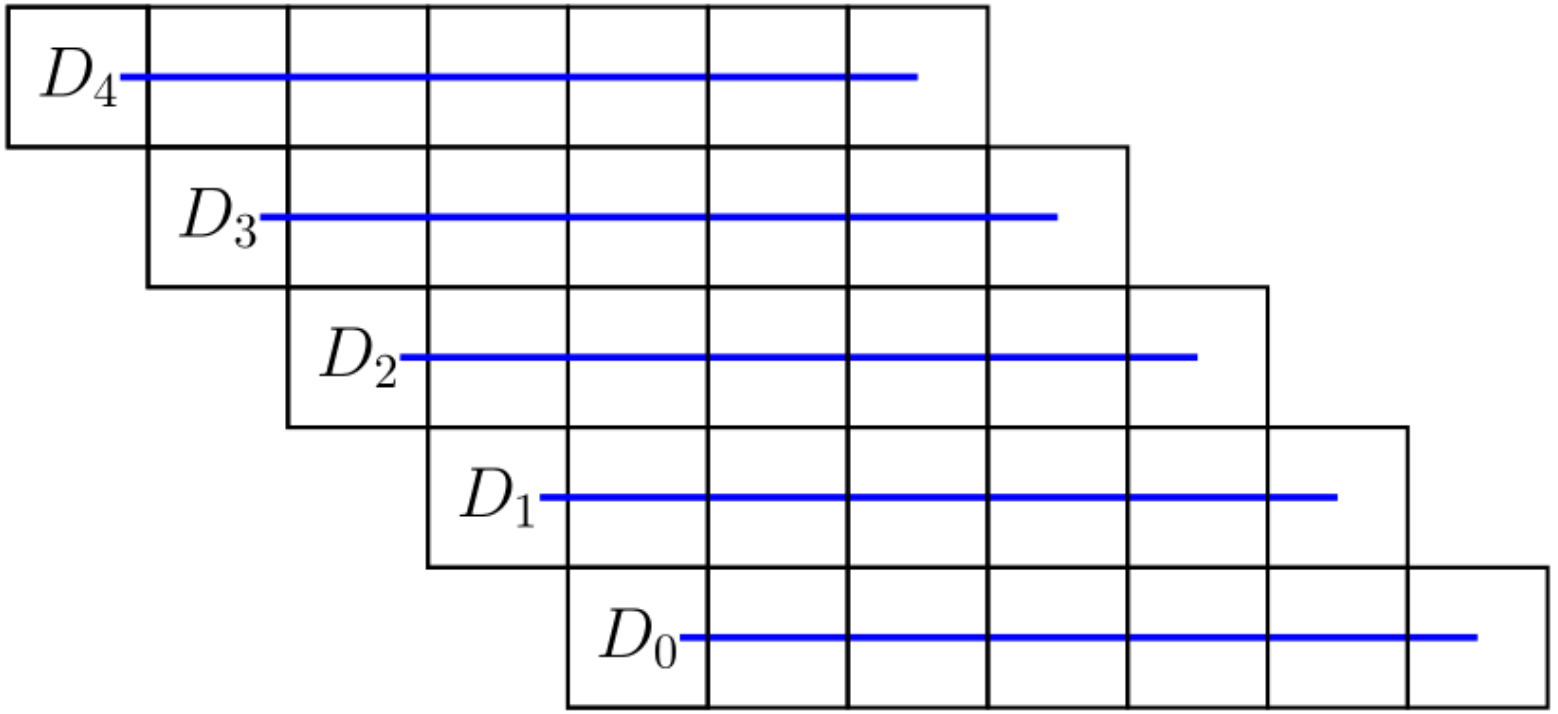
Analysis

- At most *k* times worse then the optimal algorithm



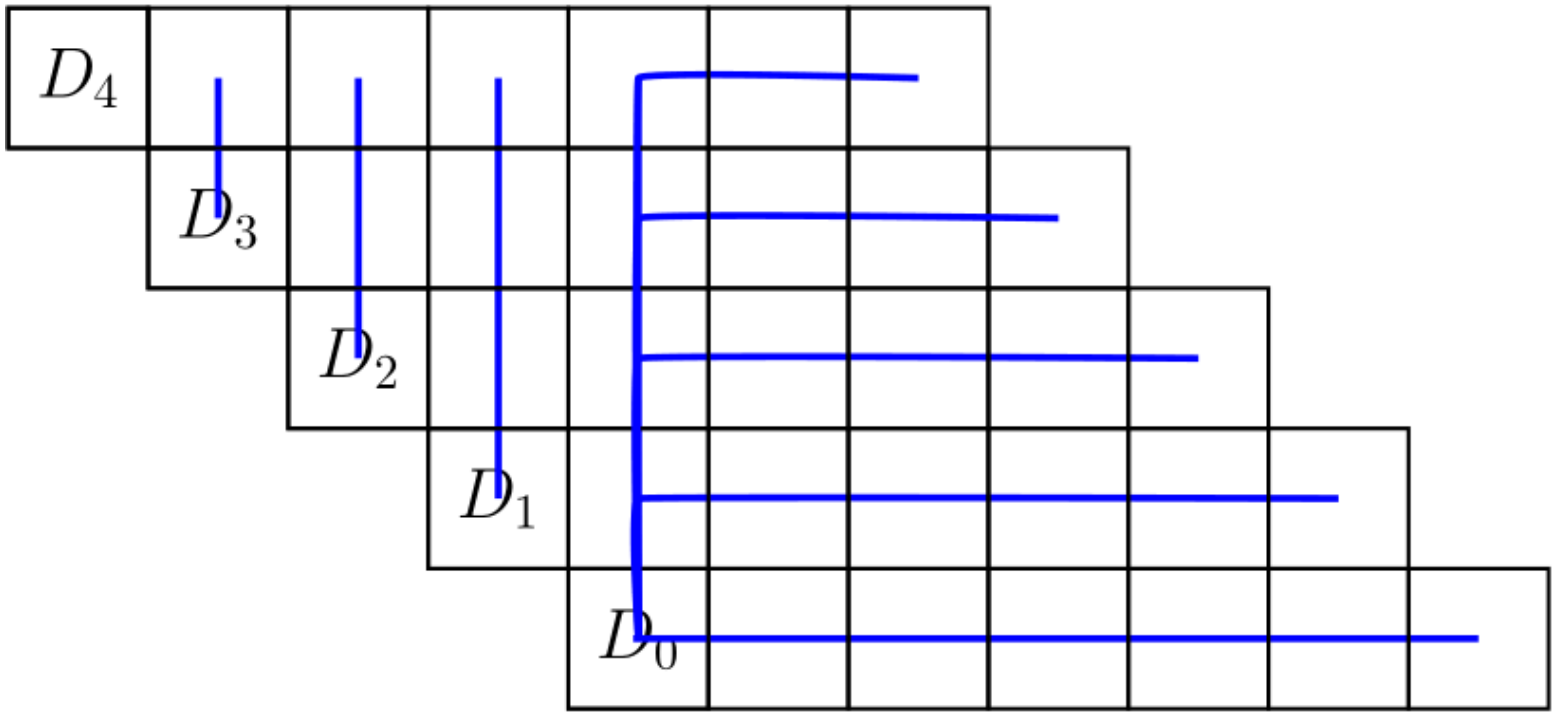
Analysis

- At most *k* times worse then the optimal algorithm



Analysis

- At most *k* times worse then the optimal algorithm



Summary

- Solve the Filling problem with robots having
  >1 hop visibility (optimal)
  >O(1) bits memory (assimp. opt)
  >O(n) rounds

- Running time
  >Assimp. opt for Single Door
  >Assimp. worst-case otpimal for Multiple Door
    – At most k times the optimal algorithm

Thank you!