Filling Arbitrary Connected Areas by Silent Robots with Minimal Visibility Range

Attila Hideg[1], Tamás Lukovszki[2], Bertalan Forstner[1]

1 Budapest University of Technology and Economics, Hungary

Attila.Hideg@aut.bme.hu, Bertalan.Forstner@aut.bme.hu

2 Eötvös Lóránd University, Budapest, Hungary

lukovszki@inf.elte.hu

Filling

- n robots with restricted capabilities
  - Limited visibility range
  - No communication
  - Limited persistent memory
- Area to fill is represented by a graph
  - Unknown, connected
  - 3D or more complex settings

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

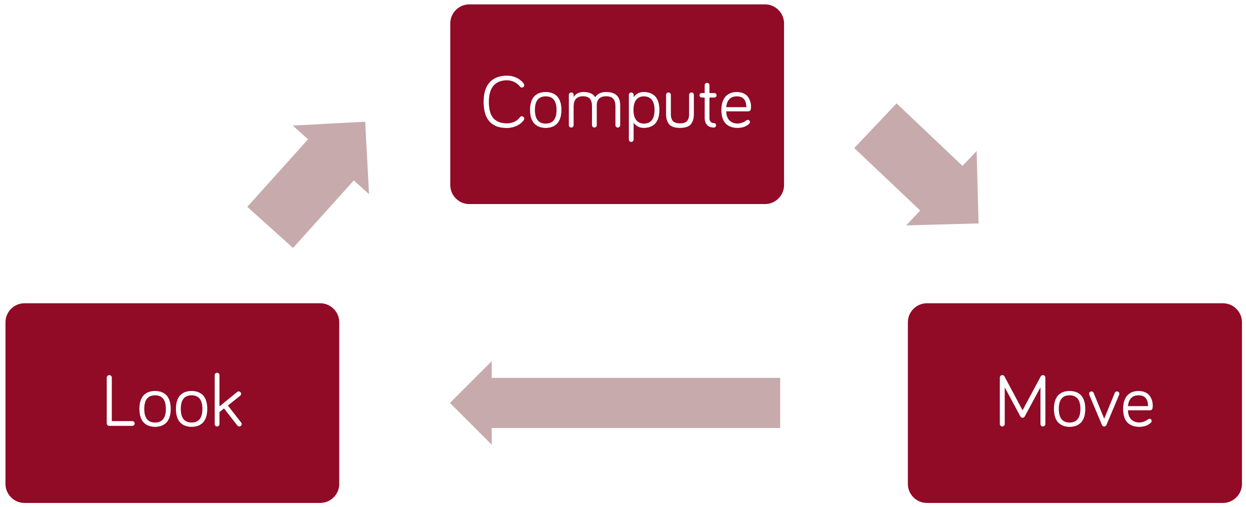
- Robots enter at the Door (or multiple Doors)
  - When a Door becomes empty, a robot is placed immediately
- They have to cover the area (graph)

Anonymous Restricted Robots

- Identical and anonymous
- Silent
- 1 hop visibility
- Limited memory
- Arbitrary graph
  - Connected
  - Fixed cyclic order of neighbors at each vertex
- Robots can move to neighboring vertices

Synchronous Look-Compute-Move (LCM)

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

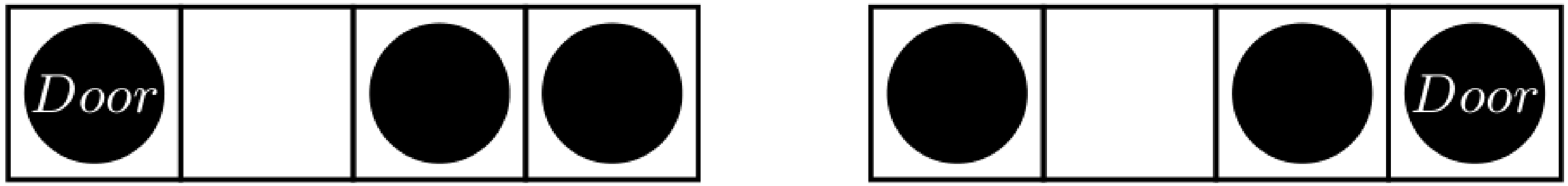


Lower bounds

- Visibility: 1 hop

- Ω(n) running time

- Memory: Ω(1) bits [Barrameda et al. 2008]



State of the art

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 [Hideg, Lukovszki 2017]	Single	1	0	13	Orthogonal
Multiple Door [Hideg, Lukovszki 2017]	Multiple	1	0	13	Orthogonal

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

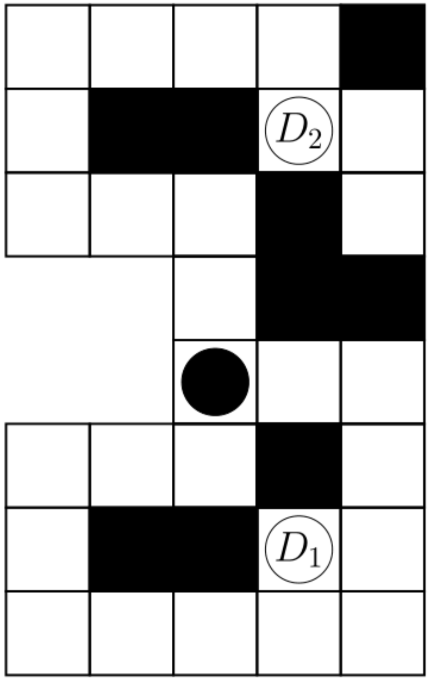
State of the art

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 [Hideg, Lukovszki 2017]	Single	1	0	13	Orthogonal
Multiple Door [Hideg, Lukovszki 2017]	Multiple	1	0	13	Orthogonal
VCM (new)	**Single**	**1**	**0**	$O(\Delta)$	**Arbitrary**
MD-VCM (new)	**Multiple**	**1**	**0**	$O(\Delta \cdot \log k)$	**Arbitrary**

Previous Result

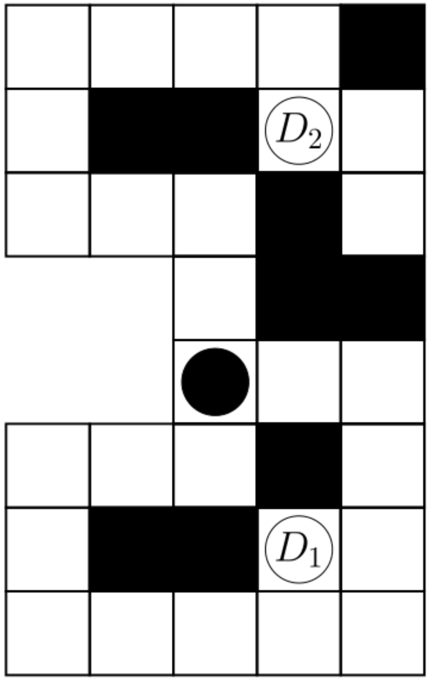
- Orthogonal

Arbitrary

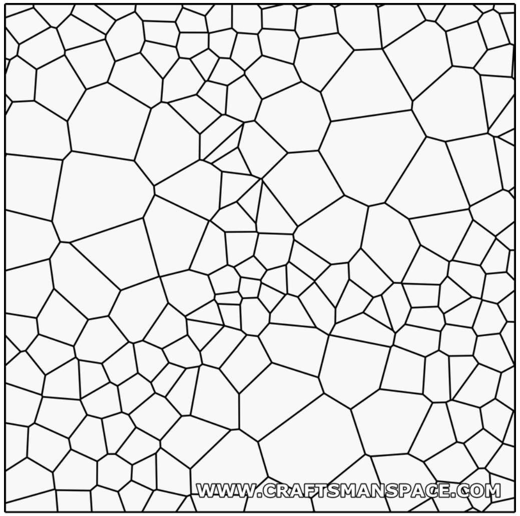


Previous Result

- Orthogonal

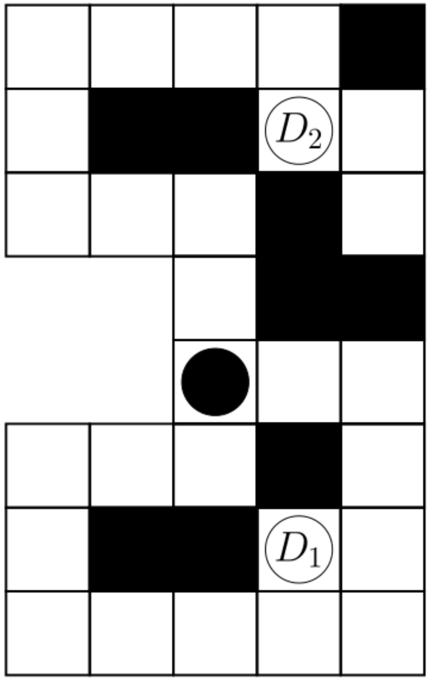


Arbitrary

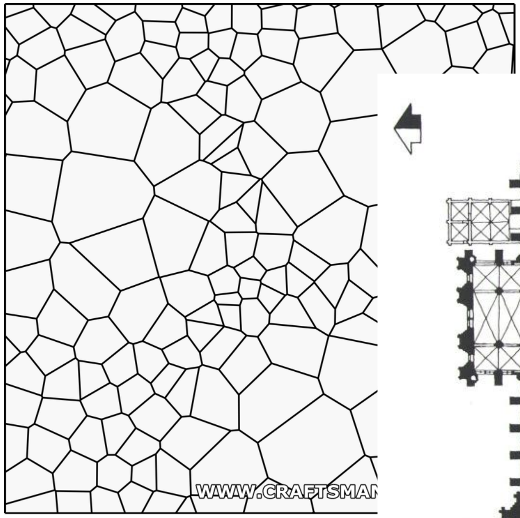


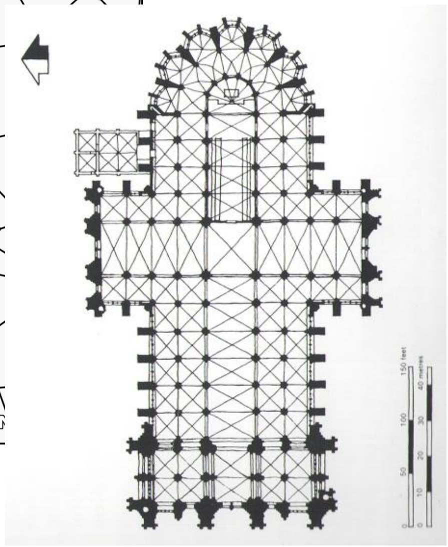
Previous Result

- Orthogonal



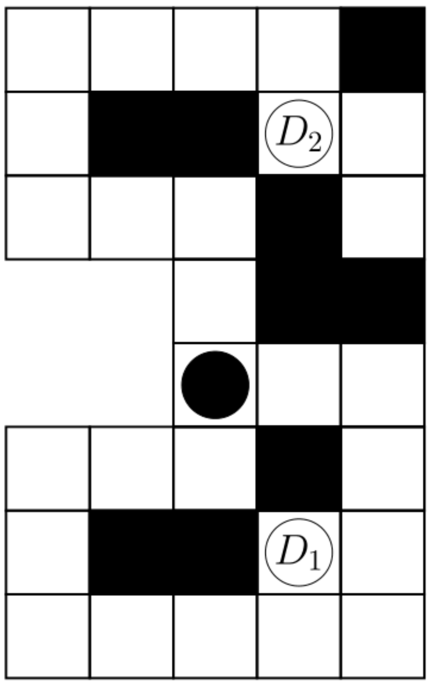
Arbitrary



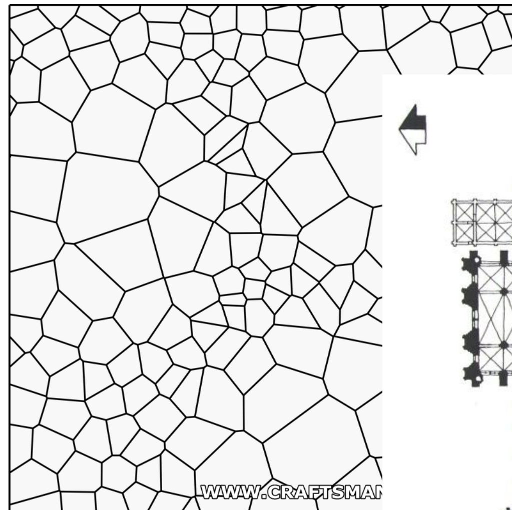


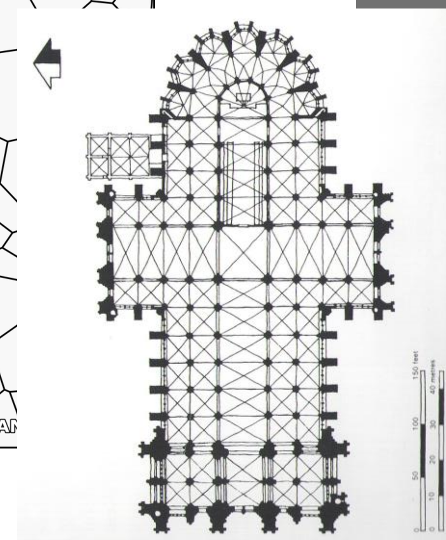
Previous Result

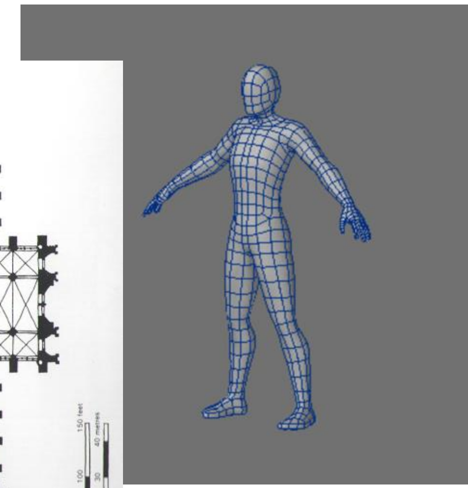
- Orthogonal



Arbitrary

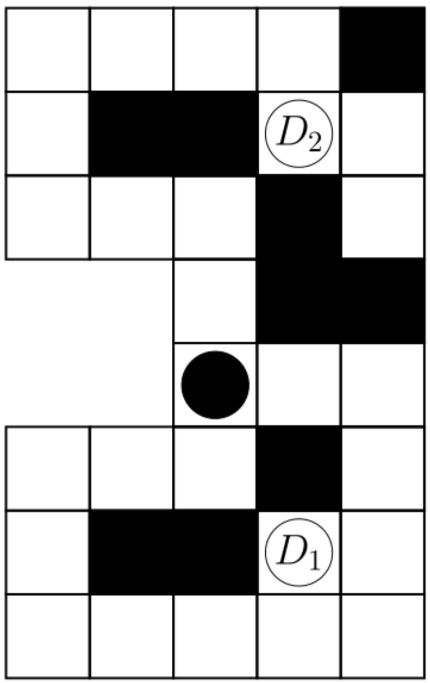




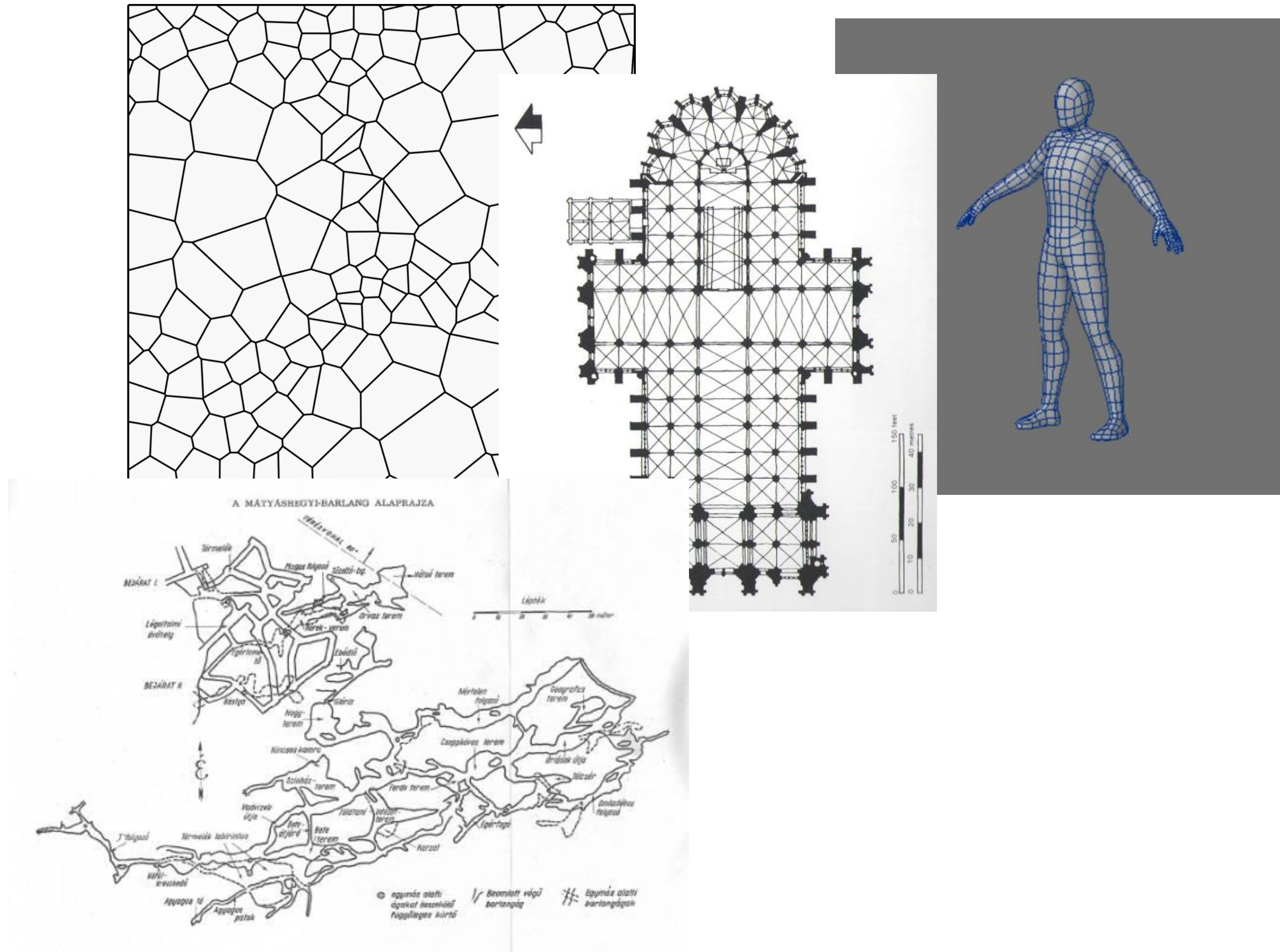


Previous Result

- Orthogonal

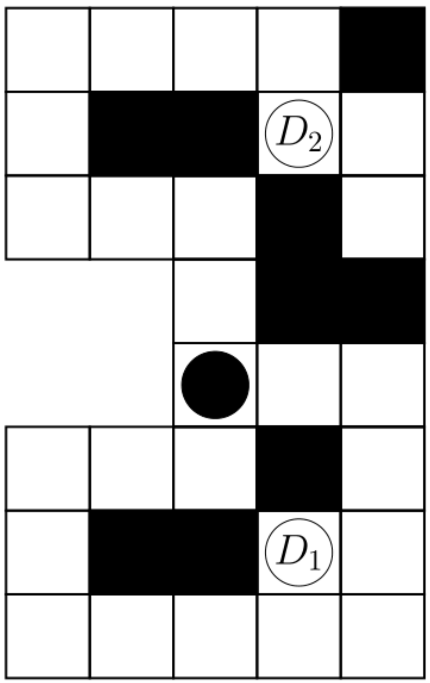


Arbitrary

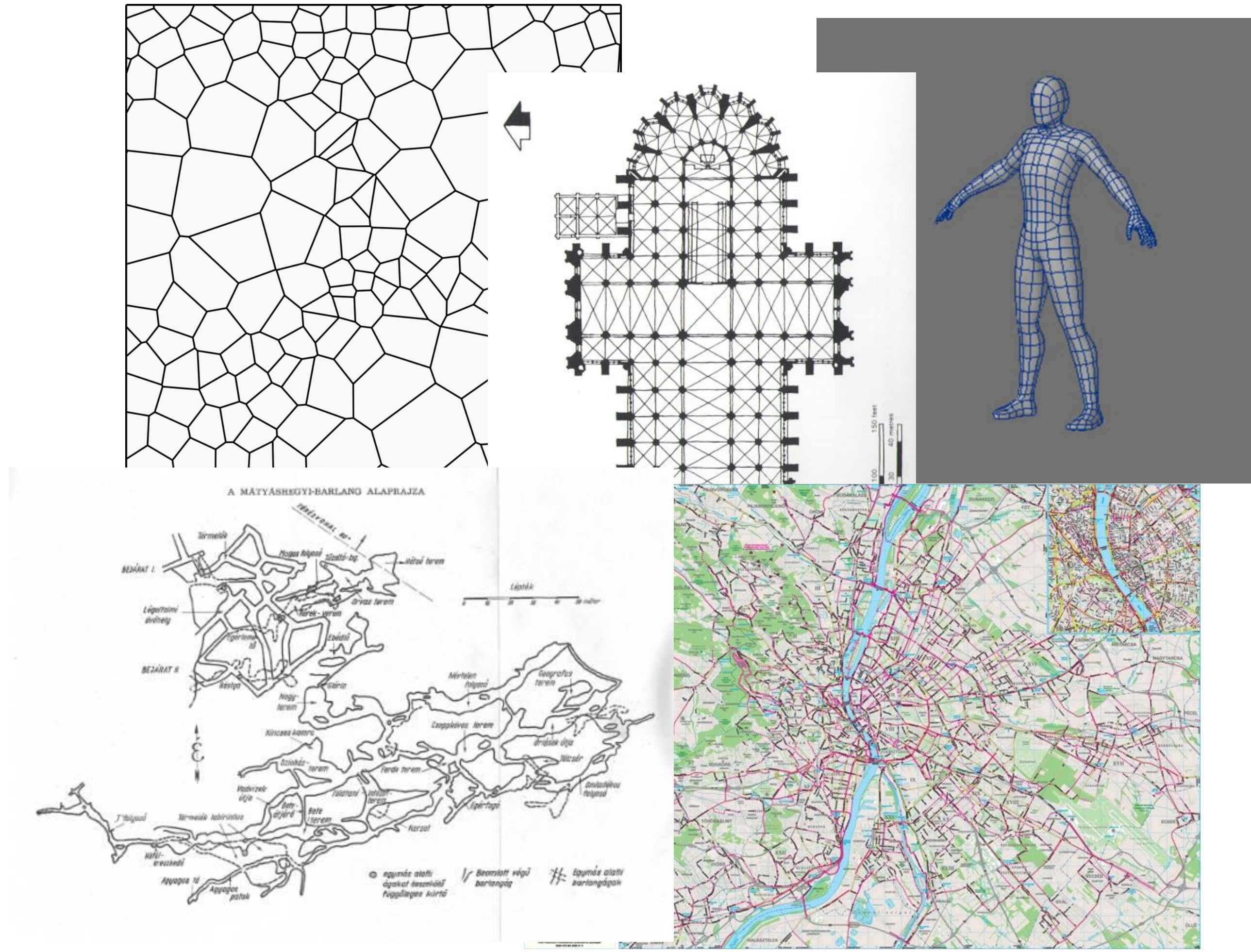


Previous Result

- Orthogonal



Arbitrary



Our Contribution (1)

- Single Door
- Requirements
  - 1 hop visibility
  - No communication
  - $O(\Delta)$ bits memory
  - Cyclic order of neighboring vertices
- Running time
  - $O(\Delta \cdot n)$ LCM cycles

Our Contribution (2)

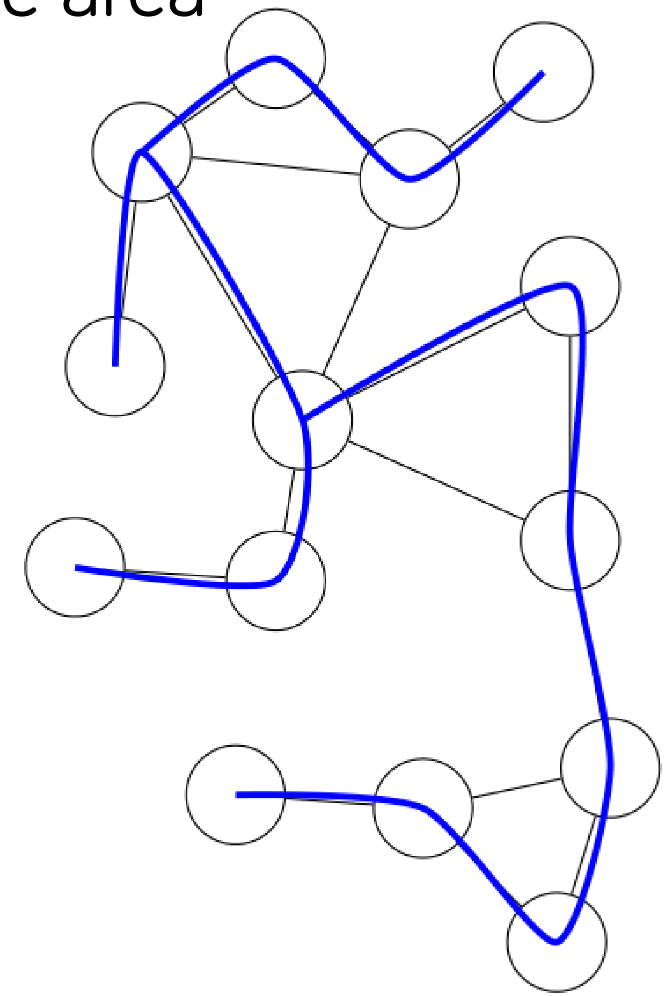
- Multiple Door
- Requirements
  - 1 hop visibility
  - No communication
  - $O(\Delta \cdot \log k)$ bits memory
  - Cyclic order for neighboring vertices
  - Robots know the index of their entry Door
- Running time
  - $O(\Delta \cdot k \cdot n)$ LCM cycles

Virtual Chain Method (VCM)

- Mimics a DFS traversal of the graph
- Main ideas:
  - Follow the Leader method
  - Timing of movements
- Chain
  - Path of the current Leader from the Door
- Main tasks to solve:
  - Prevent collisions
  - Fill the whole graph

Virtual Chain Method (VCM)

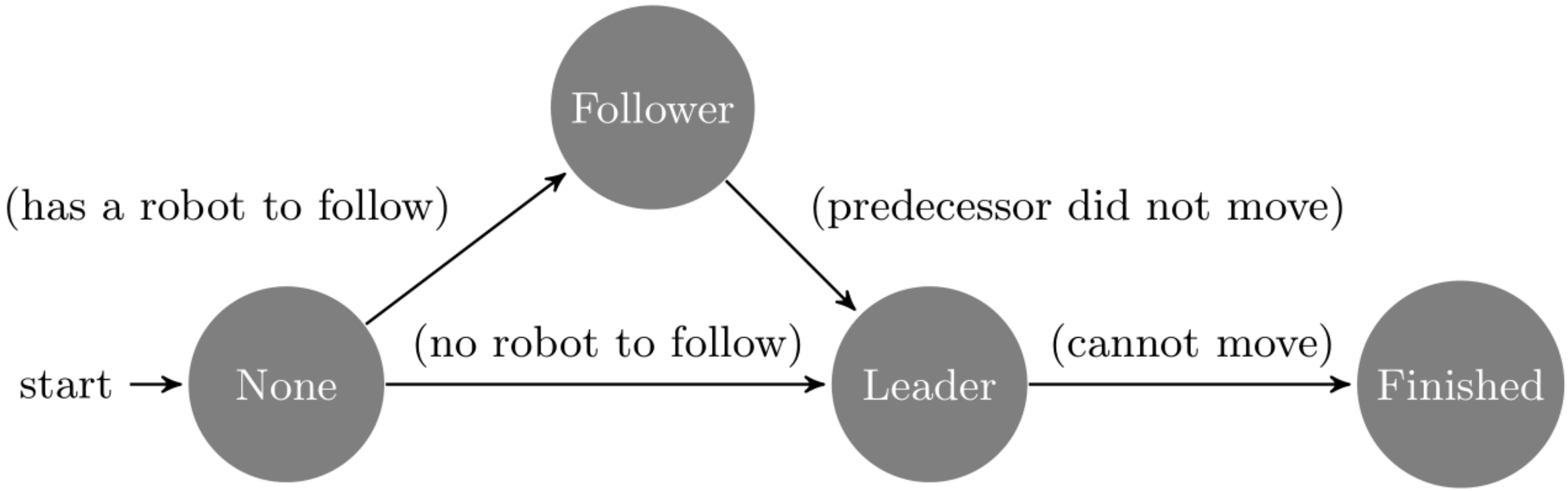
- Mimics a DFS traversal of the area
- DFS tree
  - Unknown for the robots
  - Traversed by the robots
  - Branches are filled



VCM – States

- Leader-Follower method

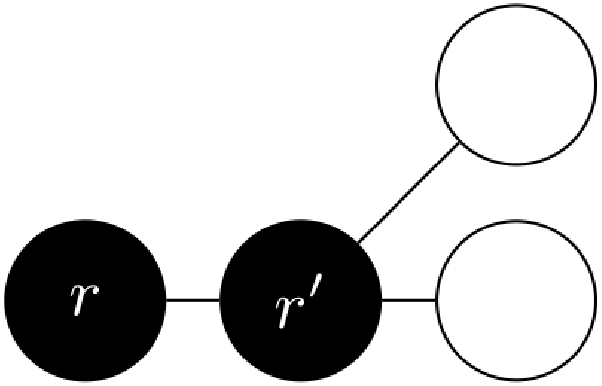
- Four possible states



VCM – Invariants

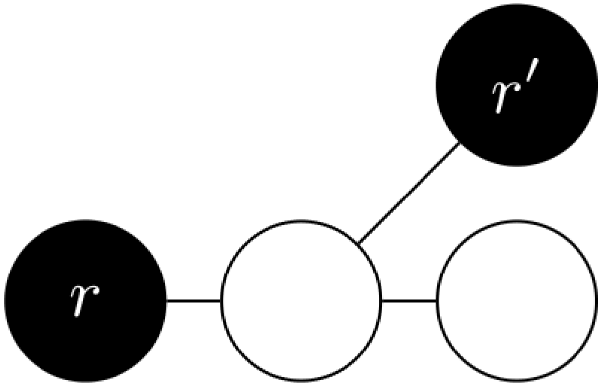
- There is at most one Leader
  - After the Leader gets stuck, the Leadership is transferred
- Followers only follow their predecessor
  - Predecessor is either in a neighboring vertex or
  - It moved away, then in the next round the follower moves to its previous position
- Each Follower has one predecessor in the chain
  - Different Followers have different predecessors

VCM – Invariants



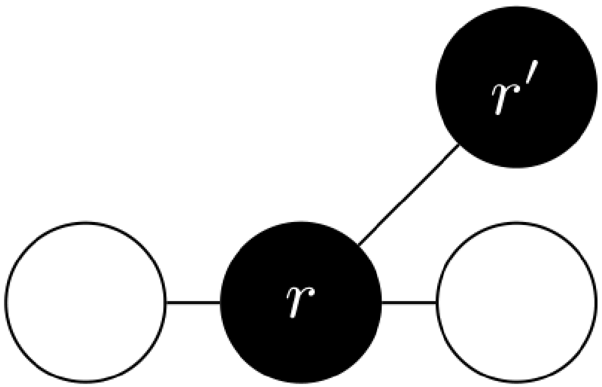
- There is at most one Leader
  - After the Leader gets stuck, the Leadership is transferred
- Followers only follow their predecessor
  - Predecessor is either in a neighboring vertex or
  - It moved away, then in the next round the follower moves to its previous position
- Each Follower has one predecessor in the chain
  - Different Followers have different predecessors

VCM – Invariants



- There is at most one Leader
  - After the Leader gets stuck, the Leadership is transferred
- Followers only follow their predecessor
  - Predecessor is either in a neighboring vertex or
  - It moved away, then in the next round the follower moves to its previous position
- Each Follower has one predecessor in the chain
  - Different Followers have different predecessors

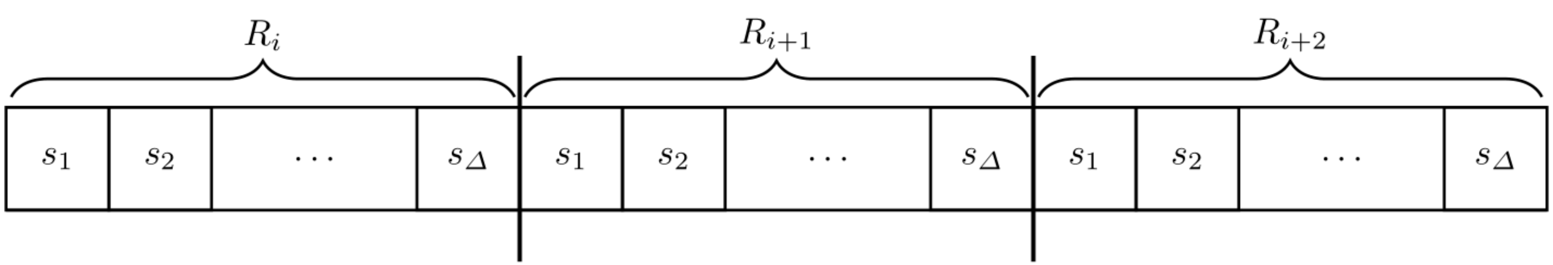
VCM – Invariants



- There is at most one Leader
  - After the Leader gets stuck, the Leadership is transferred
- Followers only follow their predecessor
  - Predecessor is either in a neighboring vertex or
  - It moved away, then in the next round the follower moves to its previous position
- Each Follower has one predecessor in the chain
  - Different Followers have different predecessors

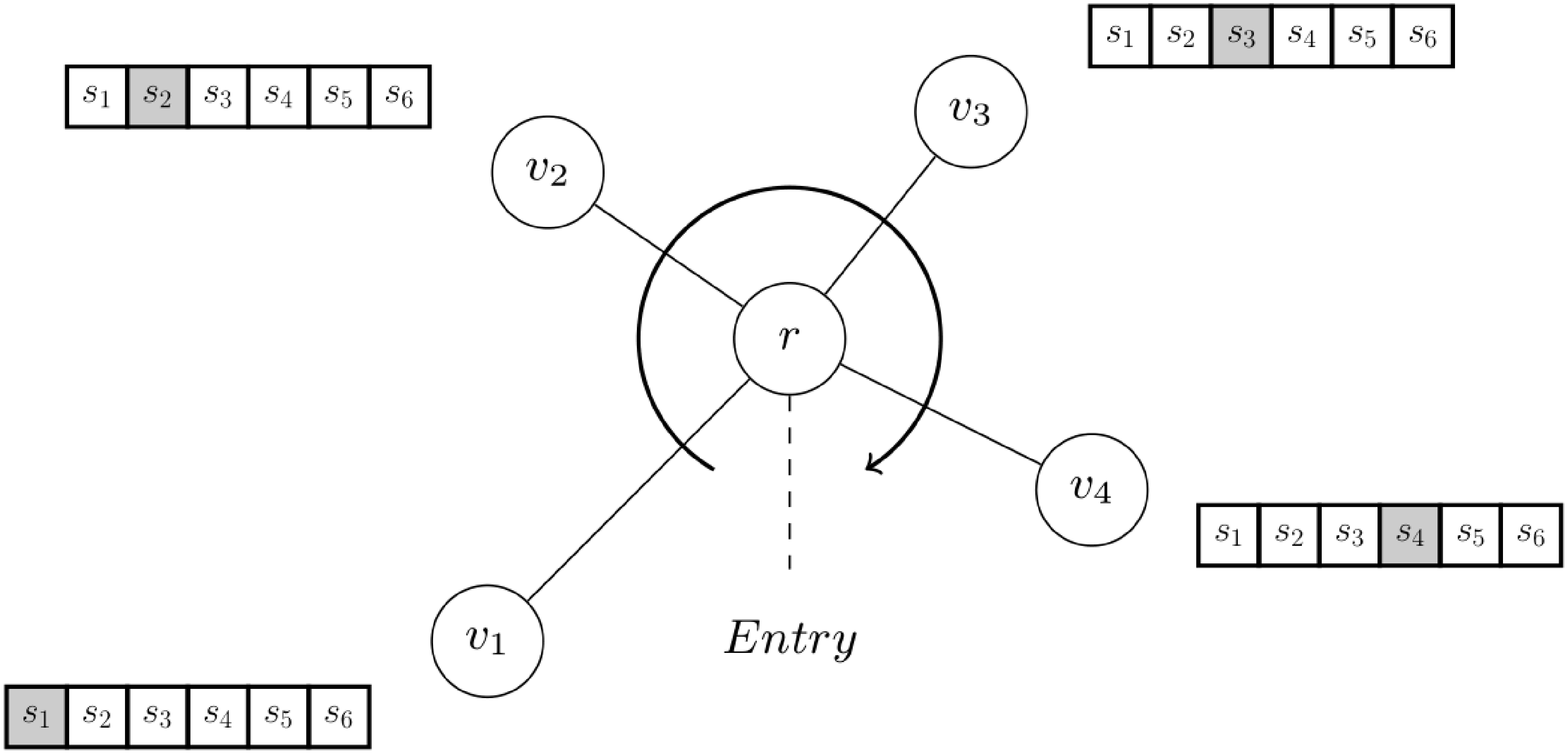
VCM – Rounds and Steps

- Timing of the movements
- Step
  - LCM cycle with an index $\mathrm{i} \in [1..\Delta]$
- Round
  - Δ steps
  - Odd and Even rounds for the robots
  - Odd: observing, Even: moving



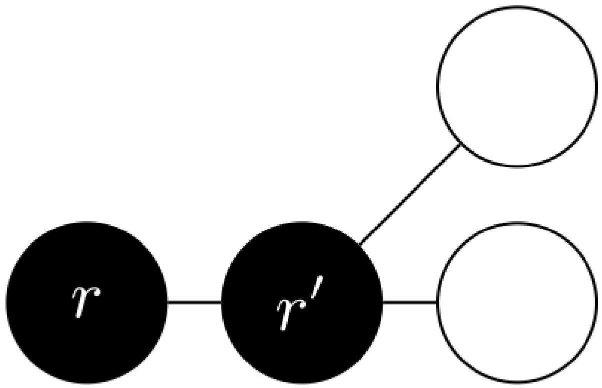
VCM – Rounds and Steps

- In the i^{th} step robots can go to the i^{th} neighbor



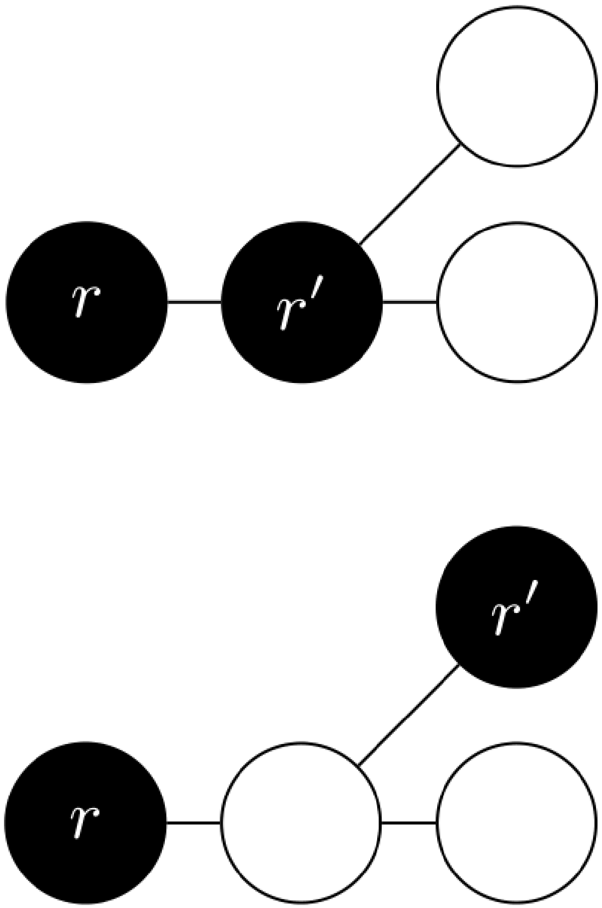
State: Follower

- Task
  - Follow its predecessor
  - One hop visibility



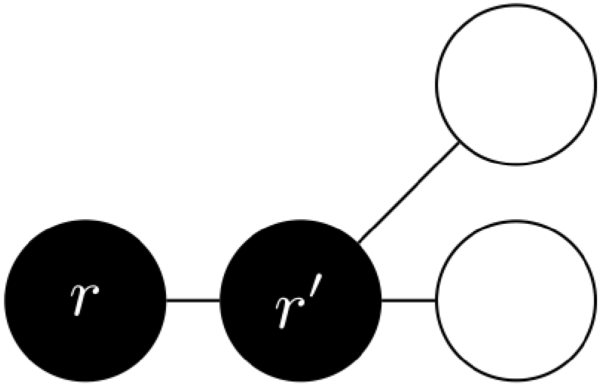
State: Follower

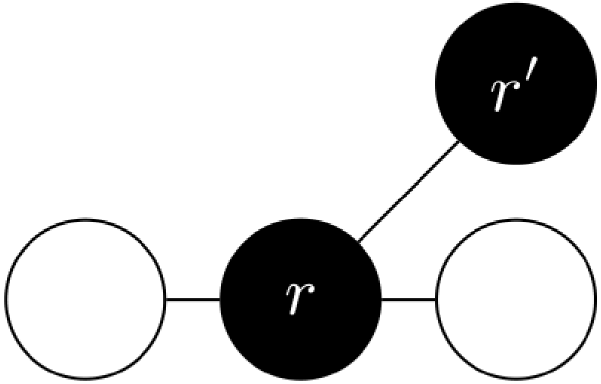
- Task
  - Follow its predecessor
  - One hop visibility
- Odd rounds
  - Observes the predecessor and empty neighbors
  - Stores which direction the predecessor moves (known from the timing!)



State: Follower

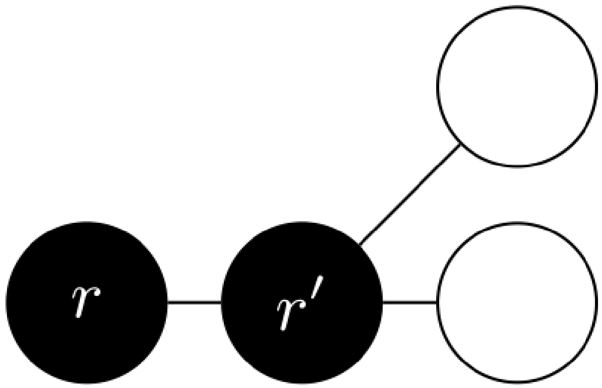
- Task
  - Follow its predecessor
  - One hop visibility
- Odd rounds
  - Observes the predecessor and empty neighbors
  - Stores which direction the predecessor moves (known from the timing!)
- Even rounds
  - Moves to the previous position of the predecessor

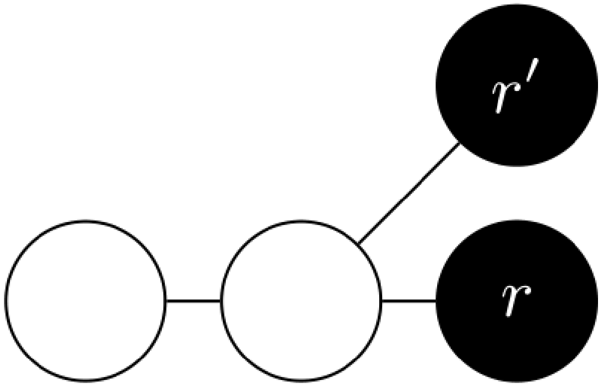




State: Follower

- 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





State: Leader

- Task
  - Has to move to an unvisited vertex (unvisited: can be decided in 2 rounds)
- Odd rounds
  - Takes a snapshot, and stores occupied vertices
- Even round
  - Moves to the neighboring vertex corresponding to the index of the step if it is unoccupied

- Switch to Finished if no unvisited vertices around

State: None

- Robot placed at the Door
  - After predecessor moved from it
  - Initial state is None
  - Assumption: degree of the Door is 1 (predecessor is at the unique neighbor)
  - Moves from Door in step S_{Δ}
  - New robot always placed in step S_1

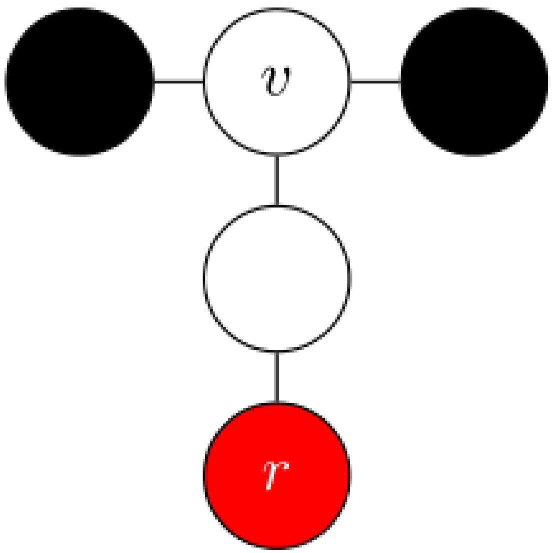
Result

Theorem 1: The VCM fills an arbitrary connected graph with a single door

- In $O(\Delta \cdot n)$ rounds
- Requirements
  - Visibility range of 1 hop
  - $O(\Delta)$ bits of persistent memory
  - Cyclic order of neighboring vertices is known at each vertex

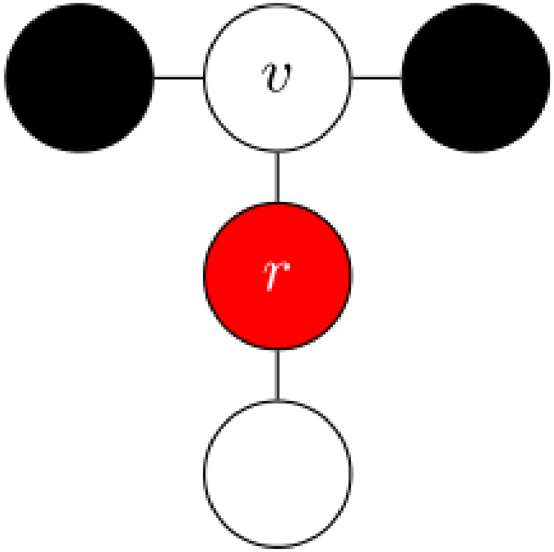
Analysis

- The Leader only moves to unvisited vertices
  - Takes snapshots in each step of the odd round
  - Knows which vertices are unvisited in the next even round



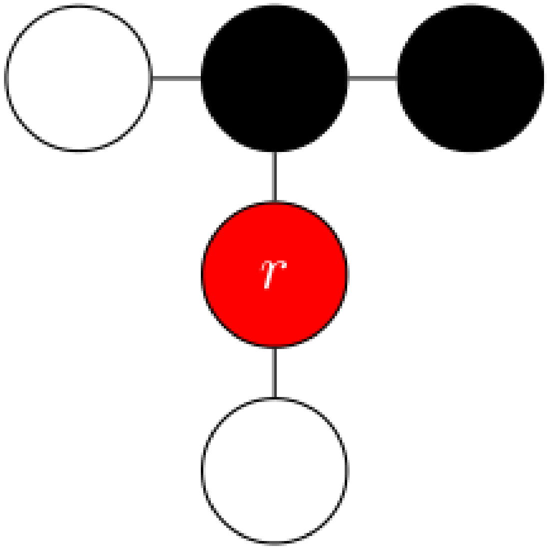
Analysis

- The Leader only moves to unvisited vertices
  - Takes snapshots in each step of the odd round
  - Knows which vertices are unvisited in the next even round



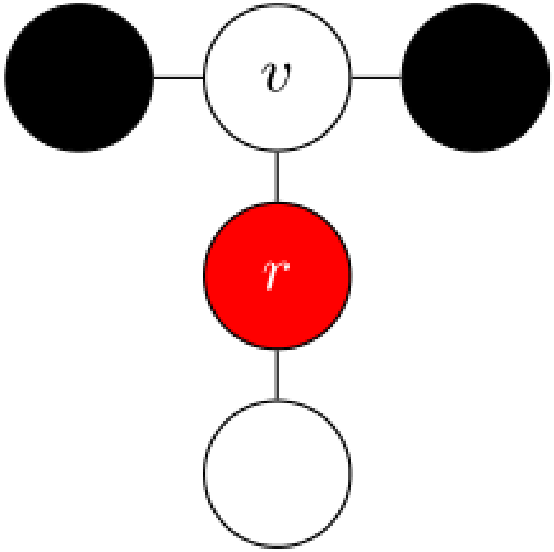
Analysis

- The Leader only moves to unvisited vertices
  - Takes snapshots in each step of the odd round
  - Knows which vertices are unvisited in the next even round



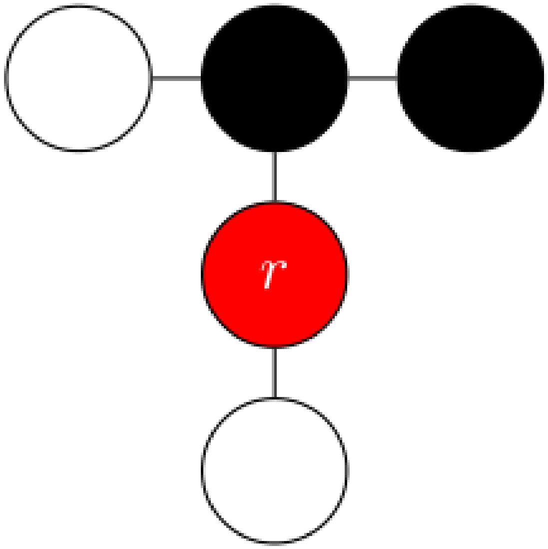
Analysis

- The Leader only moves to unvisited vertices
  - Takes snapshots in each step of the odd round
  - Knows which vertices are unvisited in the next even round



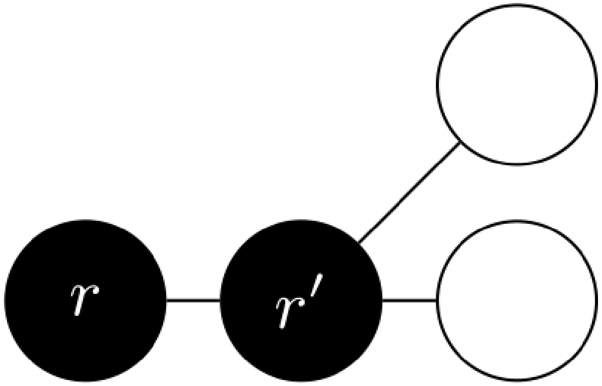
Analysis

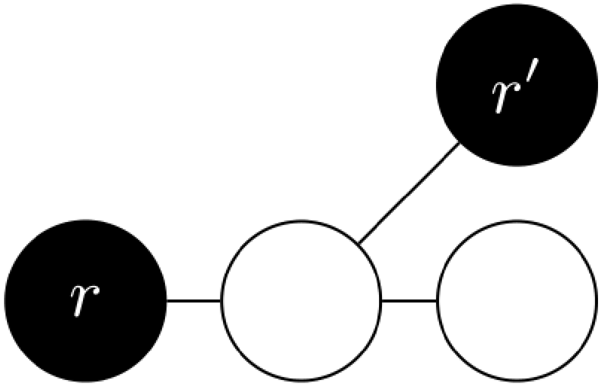
- The Leader only moves to unvisited vertices
  - Takes snapshots in each step of the odd round
  - Knows which vertices are unvisited in the next even round



Analysis

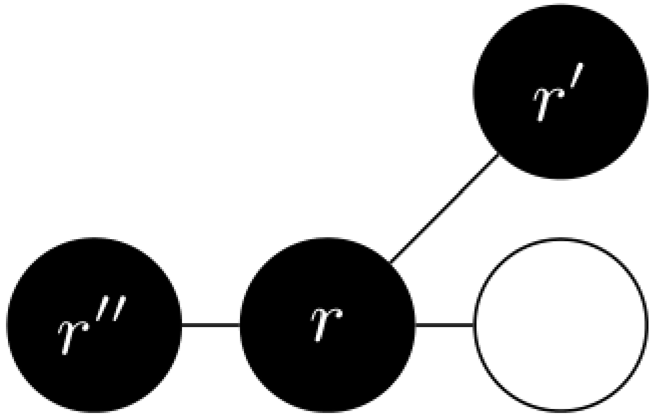
- Each Follower knows where its predecessor is
  - Odd round:
    - knows which neighbor is the predecessor
    - observes its movement
  - Even round: follows





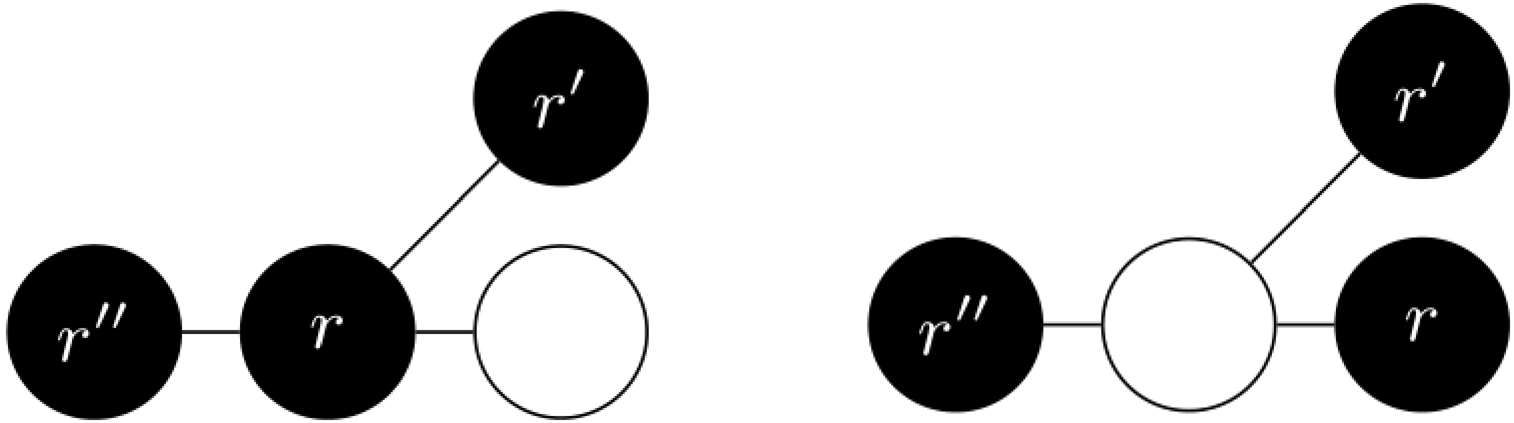
Analysis

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



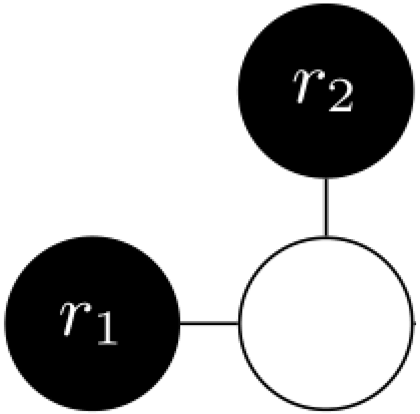
Analysis

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



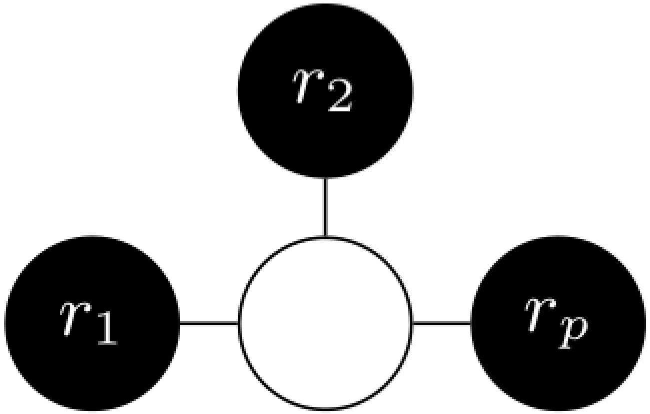
Analysis

- No collisions can occur during the dispersion



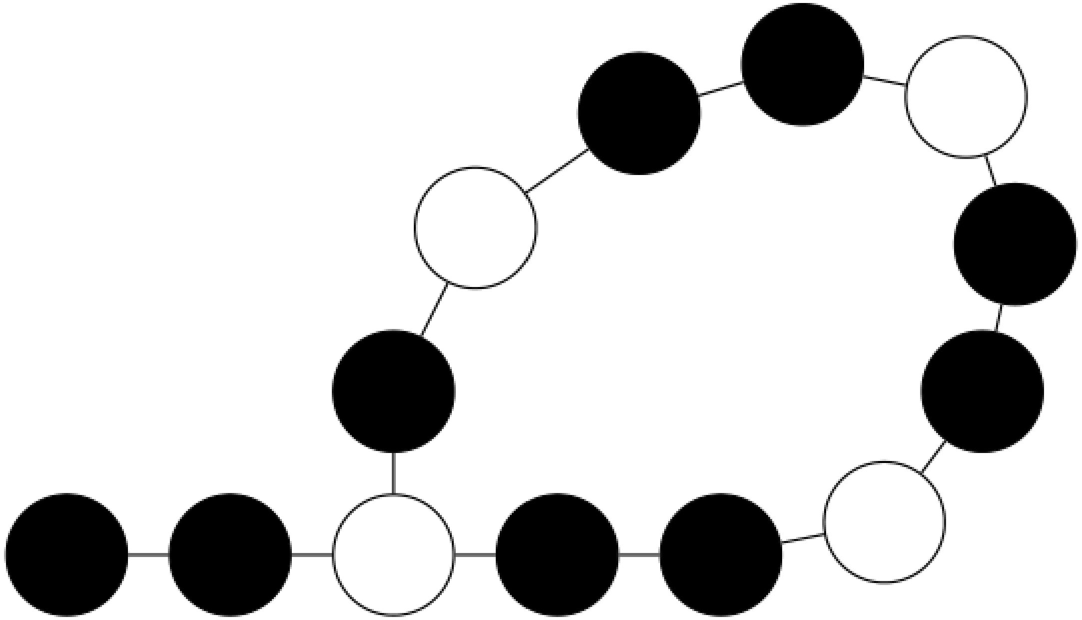
Analysis

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



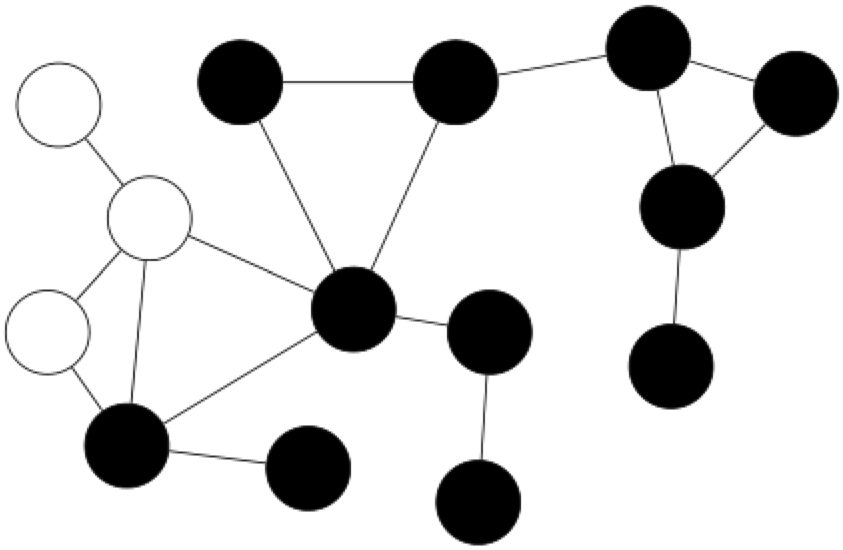
Analysis

- No collisions can occur during the dispersion
  - Follower: follows its unique predecessor
  - Leader: unvisited vertex



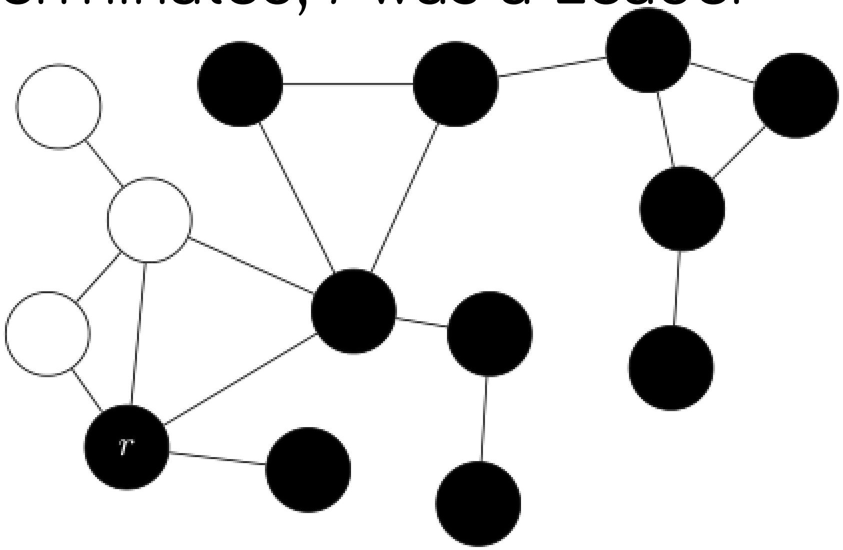
Analysis

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



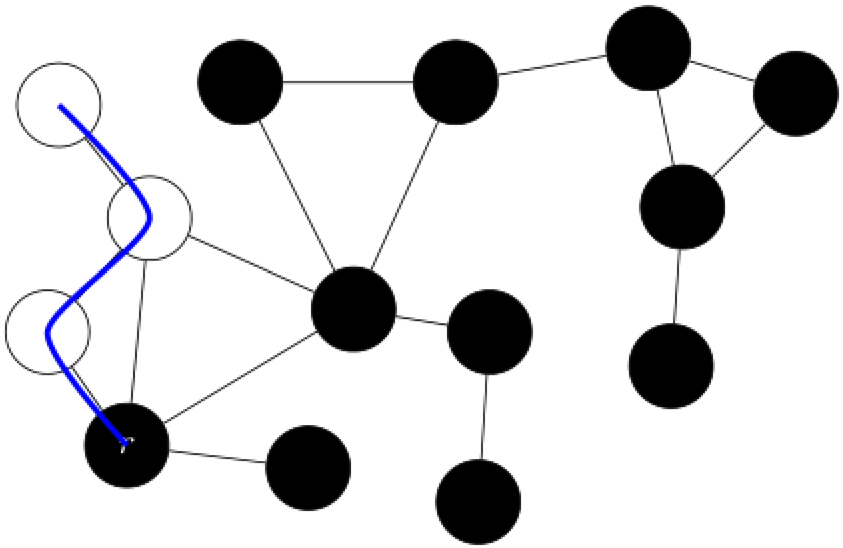
Analysis

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



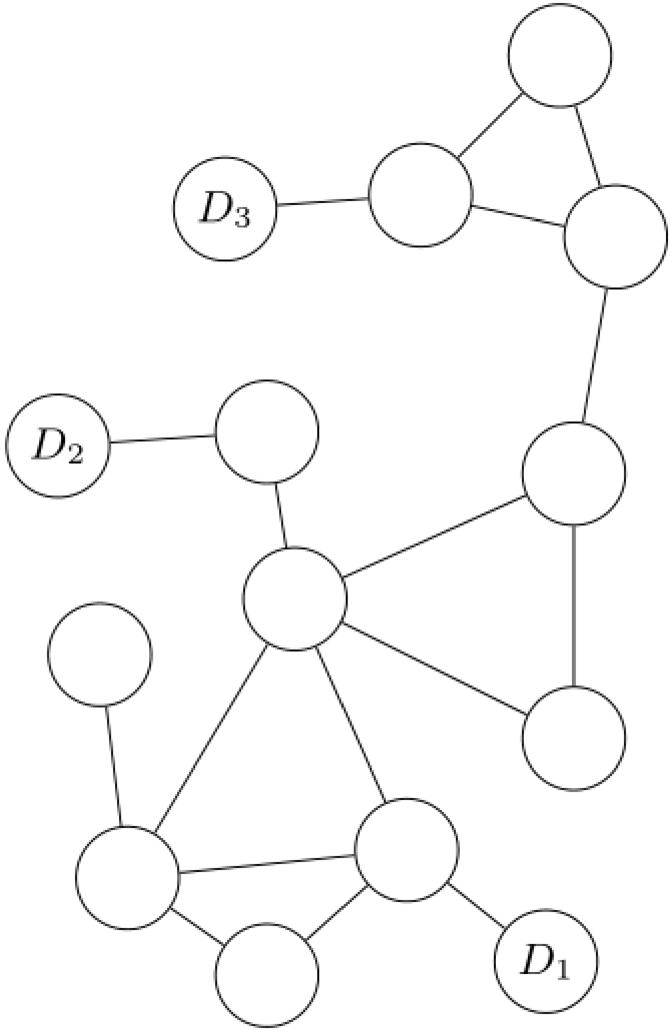
Analysis

- The proposed method fills the graph
  - Contradiction: robots cannot terminate if unoccupied vertices are present



Multiple Door VCM (MD-VCM)

- Multiple Door or k-Door filling
- Assume each Door has enough robots
- Doors have a degree of 1
    - > Two sides of a doorstep
- Different time slots are assigned to different Doors



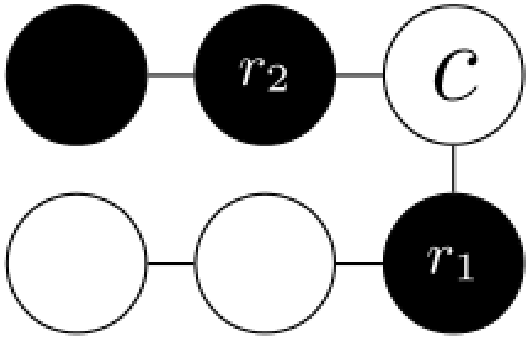
Results

Theorem 2: The MD-VCM fills an arbitrary connected graph with multiple Doors

- In $O(\Delta \cdot k \cdot n)$ rounds
- Requirements
  - Visibility range of 1 hop
  - $O(\Delta \cdot \log k)$ bits of persistent memory
  - Cyclic order of neighboring vertices
  - Degree of Door vertices are 1

Analysis

- A Leader cannot collide with another Leader



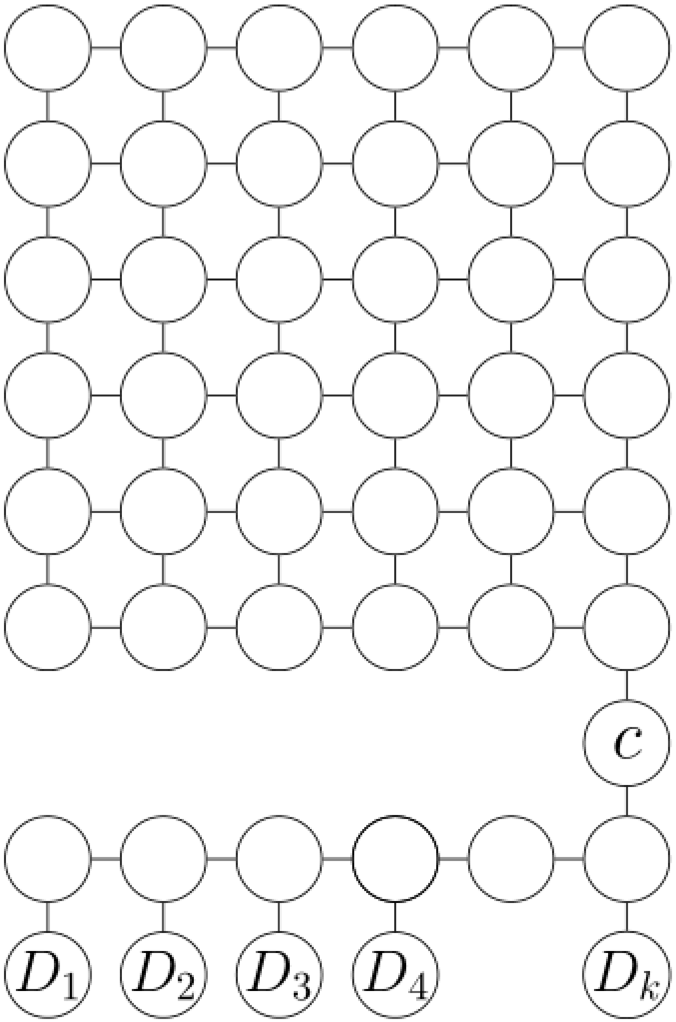
Analysis

- A Leader cannot collide with a Follower
  - Leaders can determine unvisited vertices

- Paths of different Leaders cannot cross each other
  - Leader moves to unvisited vertices
  - Leaders cannot collide

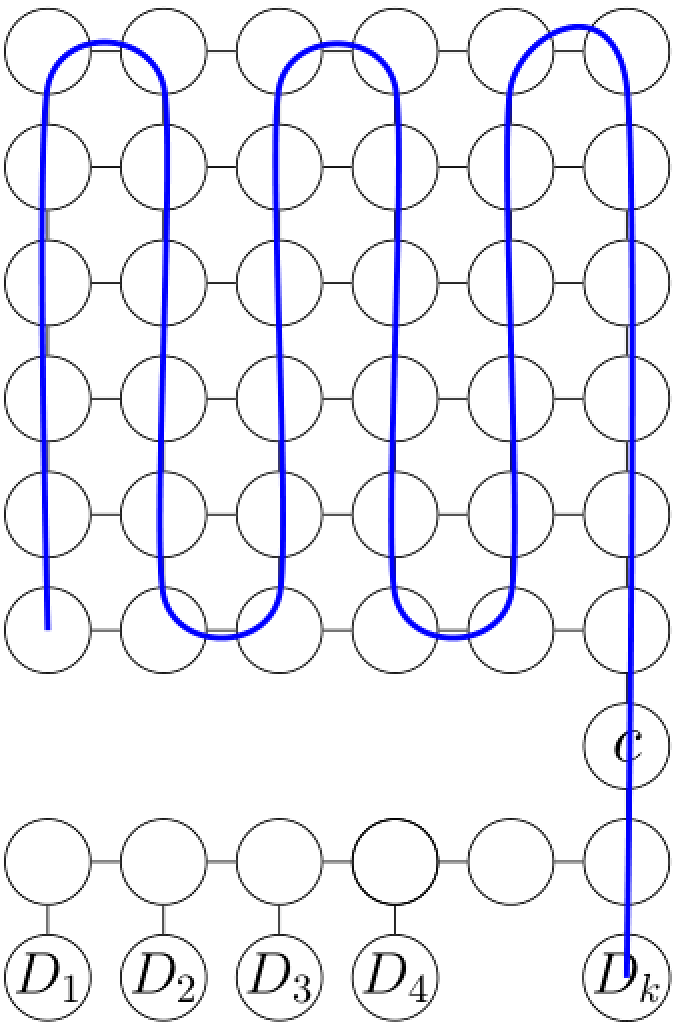
Analysis

- Worst-case
  - c is a bottleneck
  - Only robots from D_k are filling the area
  - Robots from $D_1 \ldots D_{k-1}$ are blocked by robots from D_k
- Runtime is $O(\Delta \cdot k \cdot n)$
- Optimal: $O(n)$



Analysis

- Worst-case
  - c is a bottleneck
  - Only robots from D_k are filling the area
  - Robots from $D_1 \ldots D_{k-1}$ are blocked by robots from D_k
- Runtime is $O(\Delta \cdot k \cdot n)$
- Optimal: $O(n)$



Summary

- Solve the Filling problem for arbitrary connected graphs with robots having
  - 1 hop visibility (optimal)
  - $O(\Delta \cdot \log k)$ bits memory
  - $O(\Delta \cdot k \cdot n)$ rounds
- Algorithm is simple enough to implement
  - Complex scenes
- Open question
  - Reduce the runtime (by factor of k)?

Thank you for your attention