Constrained Geometric Approximation for Robot Planning and Decentralized Formation Algorithms for Multi-Robot Systems

Yang Song
Advisor: Jason M. O'Kane

Dept. of Computer Science and Engineering
University of South Carolina

November 13, 2015

UNIVERSITY OF SOUTH CAROLINA

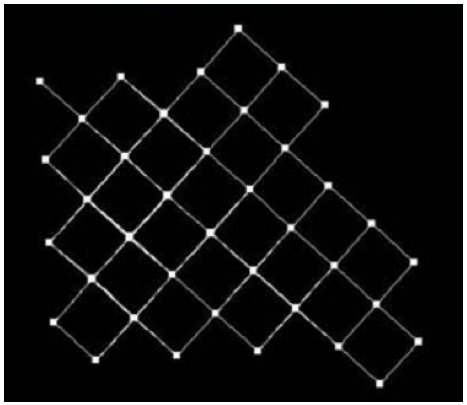
Outlines

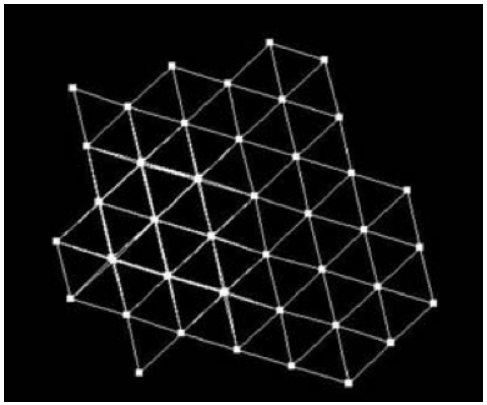
1. Distributed Multi-Robot Formation [ICRA 2014, RA-L/ICRA 2016]
   - Overview
   - Task-Assignment-based Algorithm
   - Scheduling-based Algorithm
   - Experiments
2. CGA for Robot Planning [ICRA 2012]
   - Overview
   - Contribution: Range Space
   - Experiments
3. Conclusions

Related Work

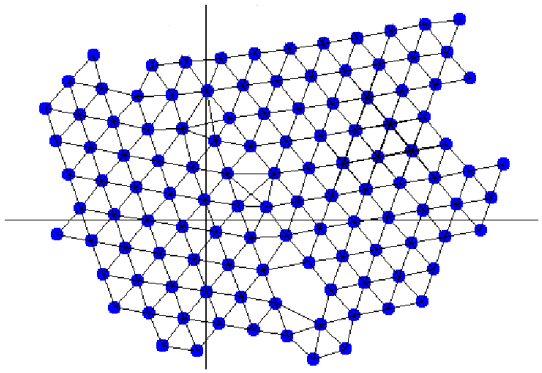
Formation using virtual force

W. Spears, D. Spears, J. Hamann and R. Heil, 2004

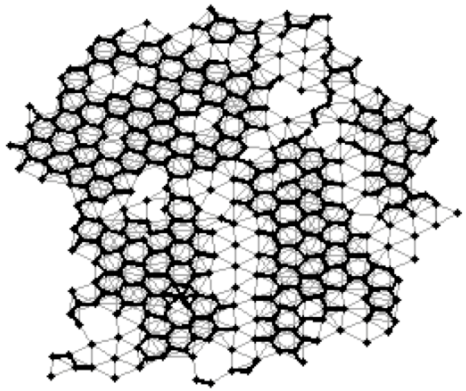




I. Navarro, J. Pugh, A. Martinoli, and F. Matia, 2008

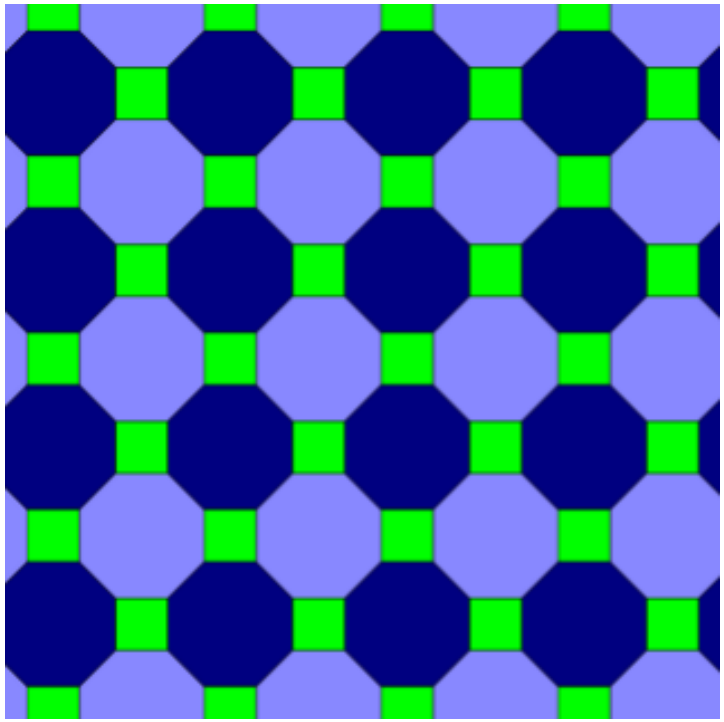


S. Prabhu, W. Li, J. McLurkin, 2012



Motivation

Question: How to use one algorithm to generate various (repeating) lattice pattern formations?

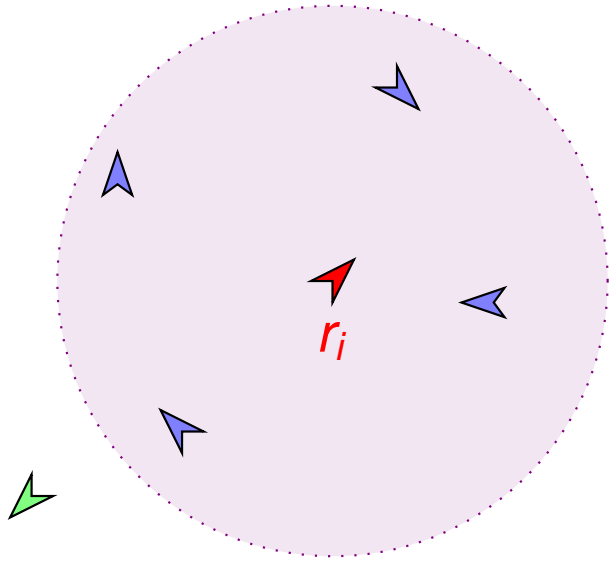


Simulation: Repeated Hexagon and Octagon-Square

100 robots

Robot Model

- Differential drive robots
- Each robot has a unique **ID**
- Each robot has a **range** within which it can sense and communicate with other robots
- Each robot gets **observation** of its neighbors' IDs and relative poses in its body frame

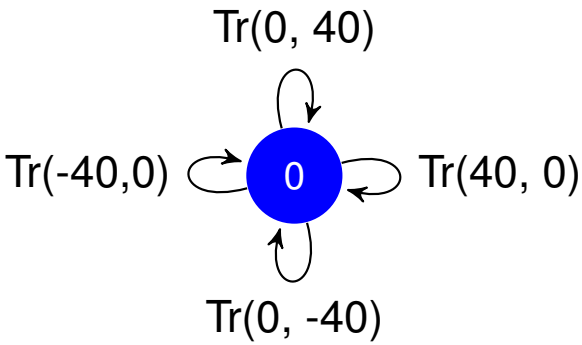


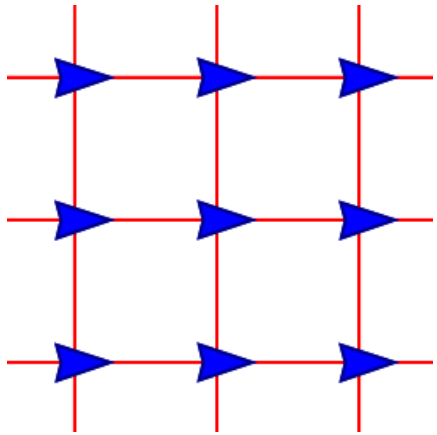
Robot r_i has 4 neighbors

Input: Lattice Graph

Definition

A **lattice graph** is a strongly connected directed multi-graph in which each edge e is labeled with a rigid body transformation $T(e)$ and each $v \xrightarrow{T(e)} w$ has an inverse edge $w \xrightarrow{T(e)^{-1}} v$.

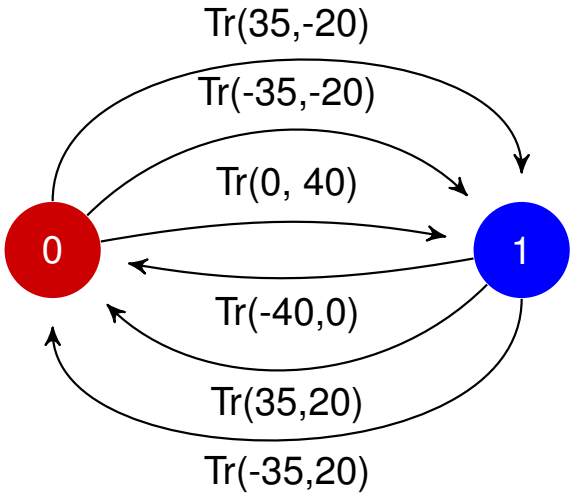


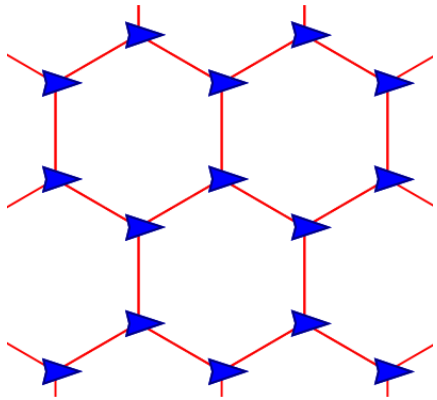


Input: Lattice Graph

Definition

A **lattice graph** is a strongly connected directed multi-graph in which each edge e is labeled with a rigid body transformation $T(e)$ and each $v \xrightarrow{T(e)} w$ has an inverse edge $w \xrightarrow{T(e)^{-1}} v$.

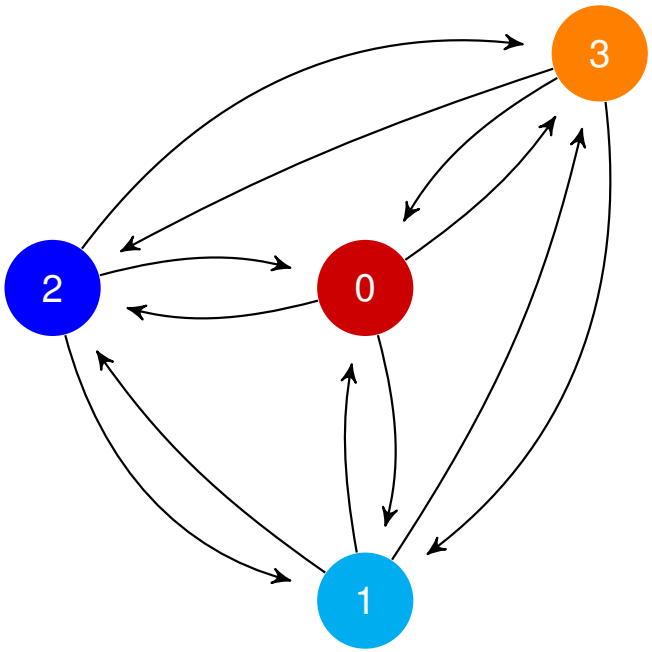


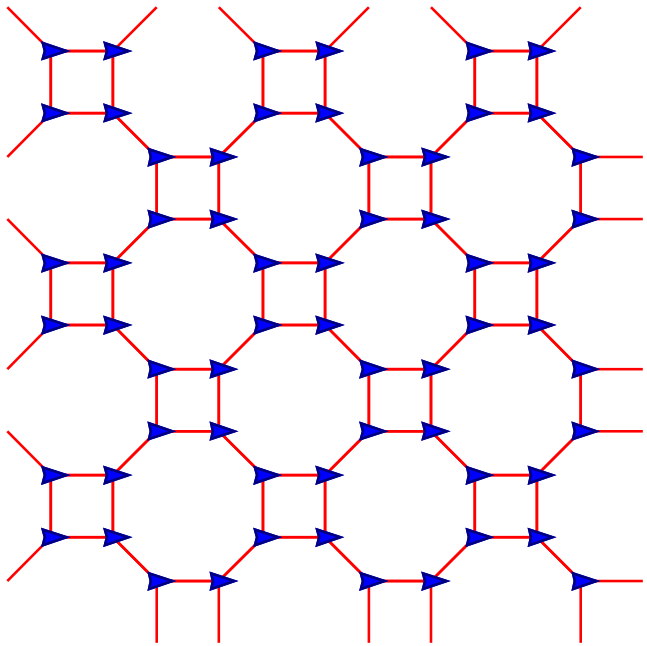


Input: Lattice Graph

Definition

A **lattice graph** is a strongly connected directed multi-graph in which each edge e is labeled with a rigid body transformation $T(e)$ and each $v \xrightarrow{T(e)} w$ has an inverse edge $w \xrightarrow{T(e)^{-1}} v$.





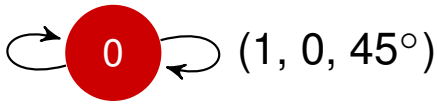
Self-Consistent Lattice Graph

Definition

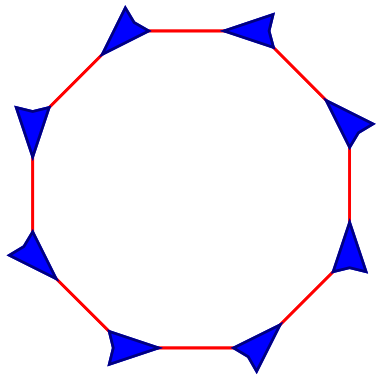
Given a range $\phi > 0$, call a lattice graph *self-consistent* for this range if, for any two paths with the same starting node,

$$v \xrightarrow{T(e_v^k)} \cdots \xrightarrow{T(e_m^w)} w, \text{ and } v \xrightarrow{T(e_v^j)} \cdots \xrightarrow{T(e_n^u)} u,$$

for which the distance between two ending nodes w and u is less than or equal to ϕ, there exist edges between w and u with right transformations.



Self-consistent Lattice Graph



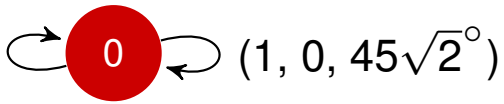
Self-Consistent Lattice Graph

Definition

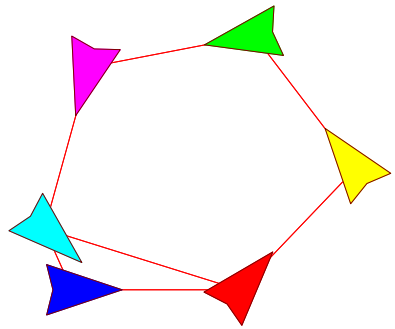
Given a range $\phi > 0$, call a lattice graph *self-consistent* for this range if, for any two paths with the same starting node,

$$v \xrightarrow{T(e_v^k)} \cdots \xrightarrow{T(e_m^w)} w, \text{ and } v \xrightarrow{T(e_v^j)} \cdots \xrightarrow{T(e_n^u)} u,$$

for which the distance between two ending nodes w and u is less than or equal to ϕ, there exist edges between w and u with right transformations.



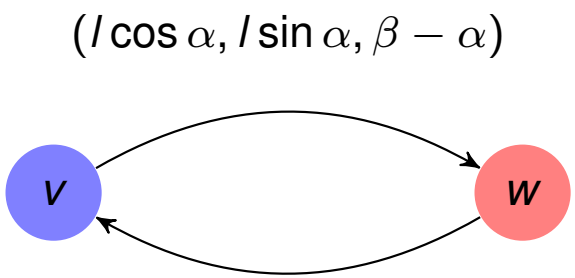
Non-self-consistent Lattice Graph



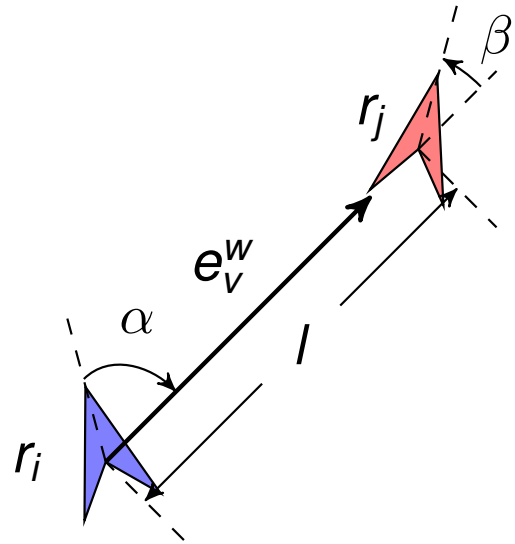
Role Function

Definition

Given a lattice graph $G = (V, E)$ and a set of robots $R = \{r_1, \ldots, r_n\}$, R **satisfies** G if there exists a role function $f : R \to V$ that preserves the neighborhood structure of G.



Two nodes in a lattice graph

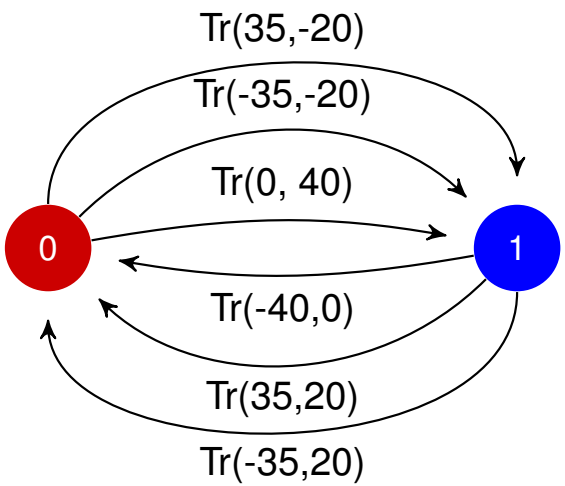


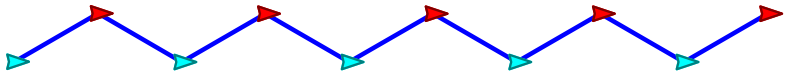
Poses of r_i and r_j satisfy the lattice graph

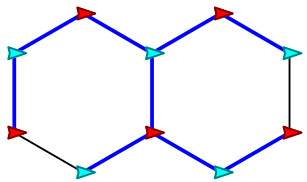
Evaluation Criteria

- **Efficiency**
  Execution time: the smallest time when all the robots reach positions and stay
- **Quality**
  Fulfillment ratio: for a robot r_i with N_i neighbors and E_i outgoing edges

$$Q = \frac{1}{n} \sum_{i=1}^{n} \frac{N_i}{E_i}$$







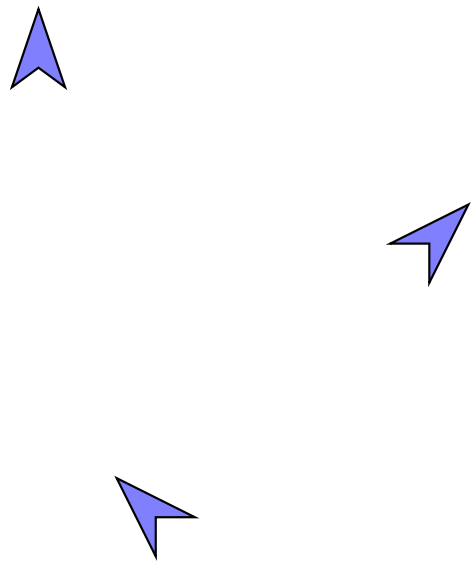
Task-Assignment-based Algorithm

General Description

Robot broadcasts message containing its

- authority
- matching

1. Form tree structure and select roles
2. Local task assignment
3. Make movement decision



Authority

Level of Importance

Definition

An **authority** is an ordered list of robot IDs

$$(\mathrm{id}_1, \ldots, \mathrm{id}_k)$$

The first ID in the list, id_1 is called the **root** ID. The final ID in the list, id_k is called the **sender** ID.

Definition

Authority A_2 is **higher than** A_1 if:

- root ID of $A_2 >$ root ID of A_1, or
- length of $A_2 <$ length of A_1 if they have the same root, or
- sender ID of $A_2 >$ sender ID of A_1 if they have the same root and length

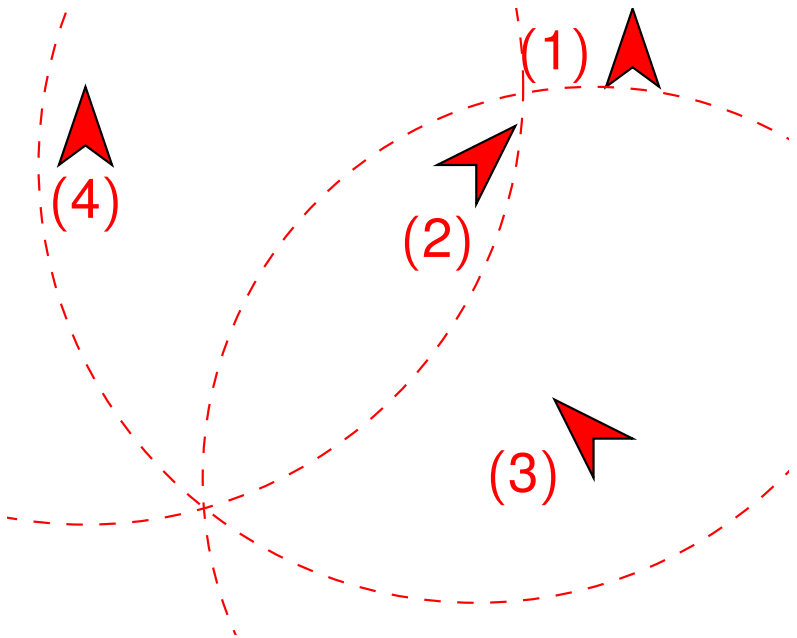
1.Construct Authority Tree

Decide to be root or descendant

Key Idea: Each robot forms an authority containing only its own ID, compares it with the authorities of remaining messages.

- Root: Robot who has the highest authority
- Descendant: Robot selects the neighbor who sends the highest authority and has matching with it as its parent
- Orphan: Robot who has no matching

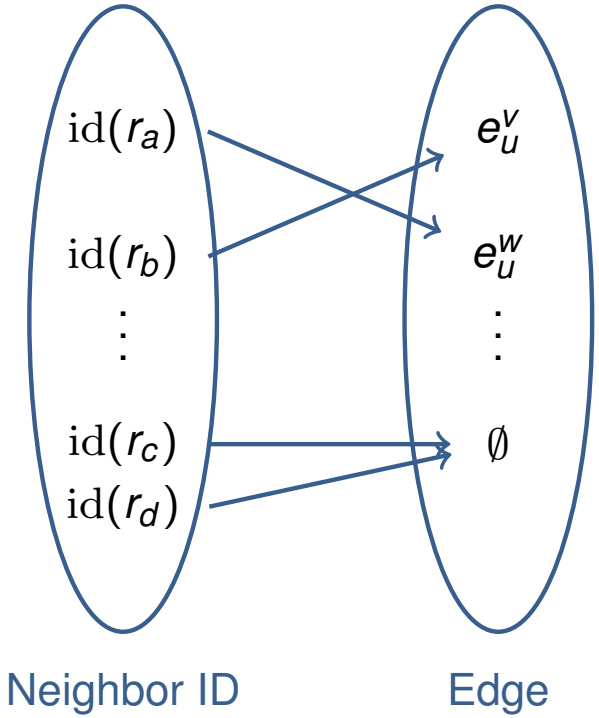
[right] Assume each robot has maximum 2 outgoing edges.



Matching

Definition

A **matching** for a robot is a function $\eta : \{\mathrm{id}(r_a), \mathrm{id}(r_b), ...\} \rightarrow \{\emptyset, e_u^v, e_u^w, ...\}$ that associates each neighbor ID with either a lattice graph edge from its role vertex or with the null value $\emptyset$.

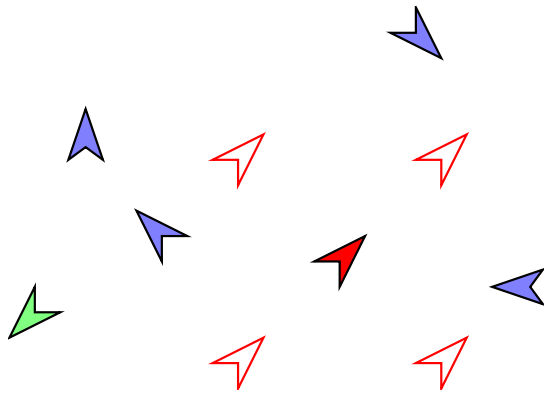


2. Local Task Assignment

Hungarian Algorithm

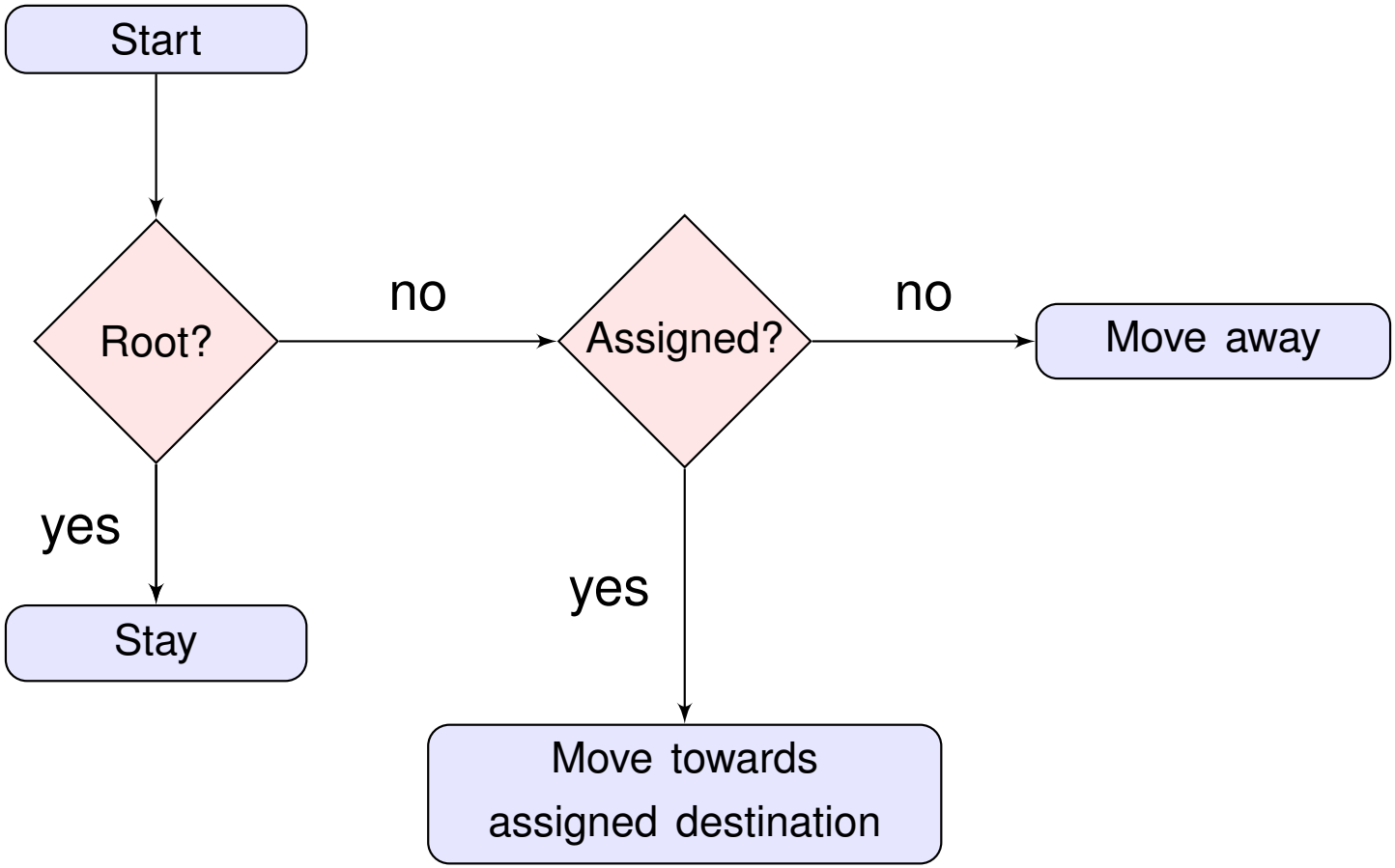
To compute an optimal matching of a robot with N neighbors and E out-going edges of its role in the lattice graph, define a weight matrix of size $\min(N, E) \times E$ and apply Hungarian Algorithm (Harold W. Kuhn, 1955).

1. Each row corresponds to a neighbor
2. Each column corresponds to an out-going edge of robot's role
3. The entries of the matrix are the Euclidean distance between current position of each neighbor and the desired position if matched with a lattice graph edge



5 neighbors, 4 out-going edges.

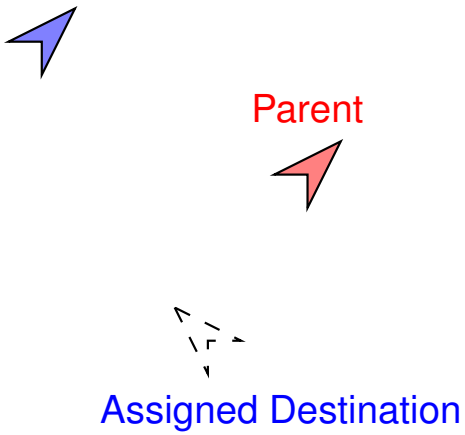
3. Robot Movement Strategy



Bounded Movement

Goal: Keep descendant in the range of its parent

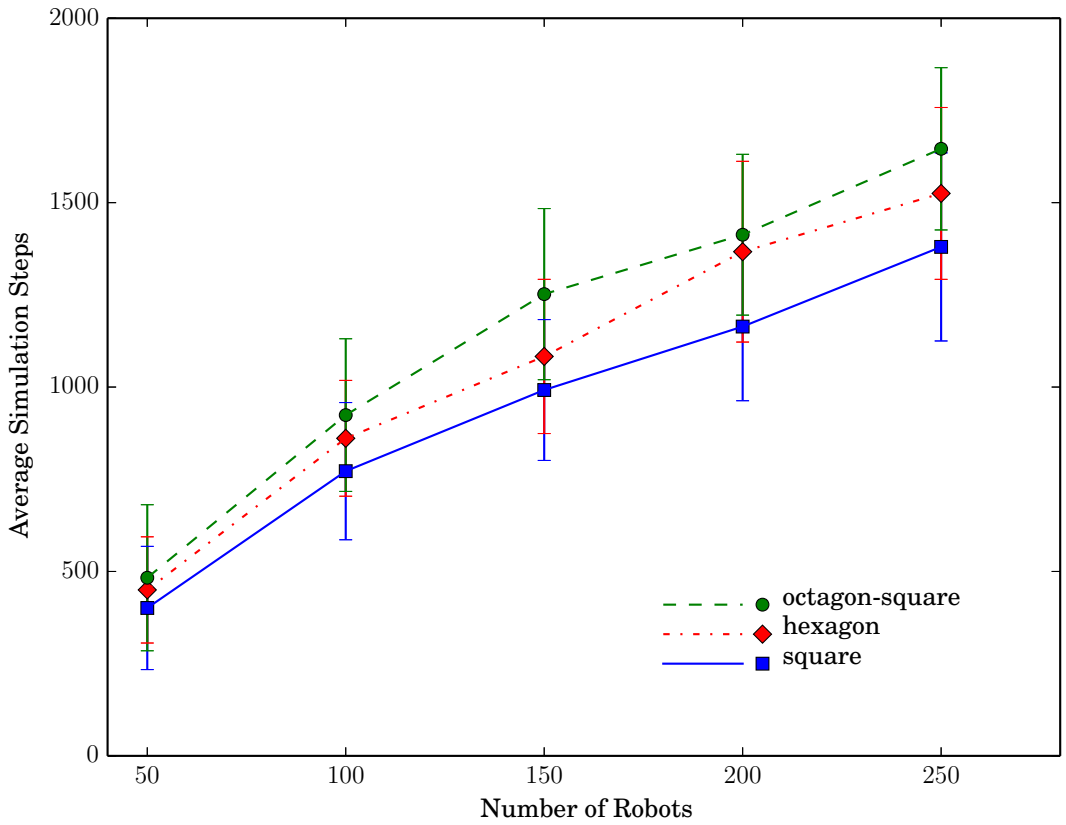
- Descendant can reach anywhere in set P (blue circle) at next stage
- Descendant is guaranteed to be observed in set O (Red circle) by its parent at next stage
- The real destination for descendant is the closest point on the boundary of the intersection ($O \cap P$) to the assigned destination



Simulation

We run simulations of 3 repeating patterns: square, hexagon, octagon-square.

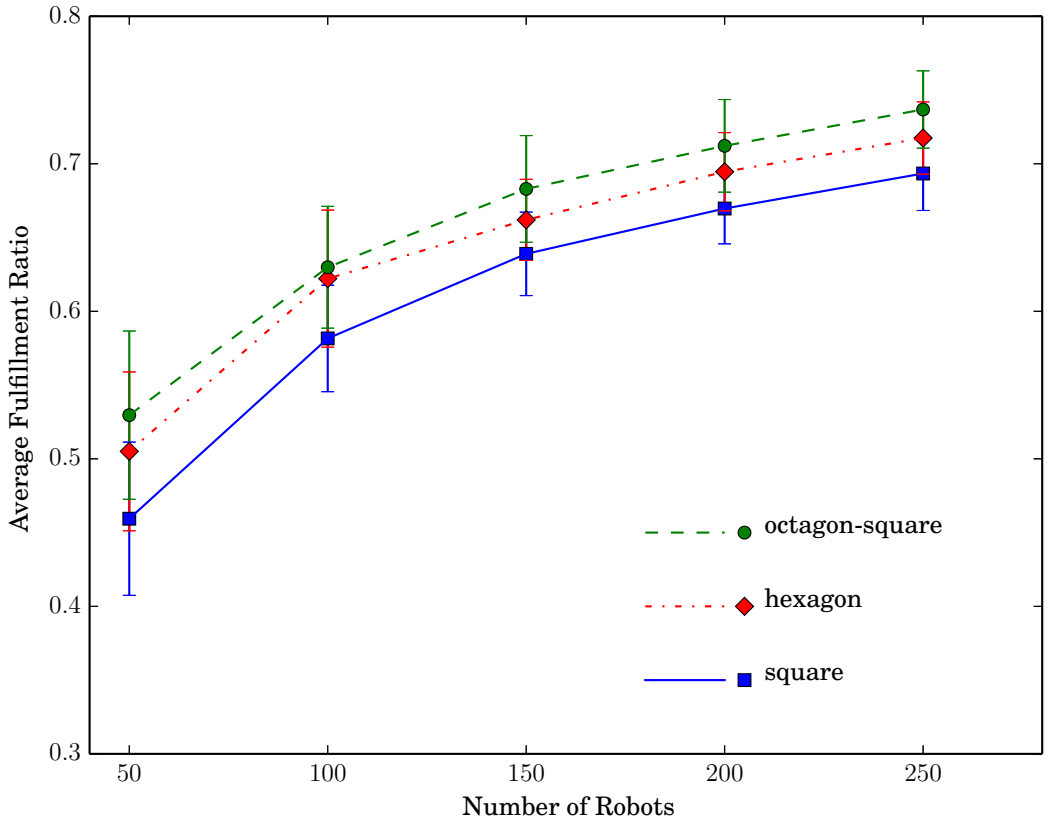
1. For each lattice pattern, the number of robots $n = 50, 100, 150, 200, 250$.
2. For each n, we run 50 trials with random initial poses.



Simulation

We run simulations of 3 repeating patterns: square, hexagon, octagon-square.

1. For each lattice pattern, the number of robots $n = 50, 100, 150, 200, 250$.
2. For each n, we run 50 trials with random initial poses.



Simulation: Robust Test

Square pattern, 100 reducing 1/4 robots

Summary: Advantages and Limitations

Advantages:

1. Scales well with increasing numbers of robots.
2. Stateless and robust to system failure.

Limitations:

1. Cannot guarantee to finish in finite time (frequent authority tree destruction).
2. Cannot guarantee formation quality.

Simulation: Repeated Square

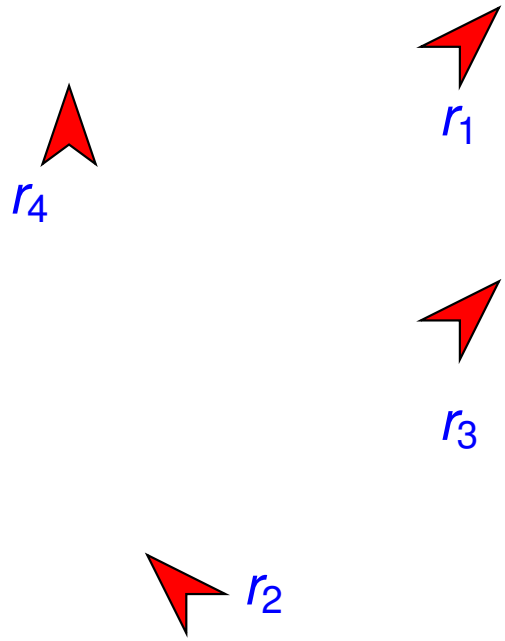
32 robots

Scheduling-based Algorithm

Key: Sequentially drives a robot in a decreasing order of their IDs to a vacancy until no more un-scheduled robots.

Robot broadcasts message containing:

- HighestID
- Root: HighestID among all Stable, Vacancy
- Vacancy
- Relocate: HighestID among all Unstable
- Subtree: if a descendant of the Relocate
- Parent
- Path to Root
- Path to Relocate

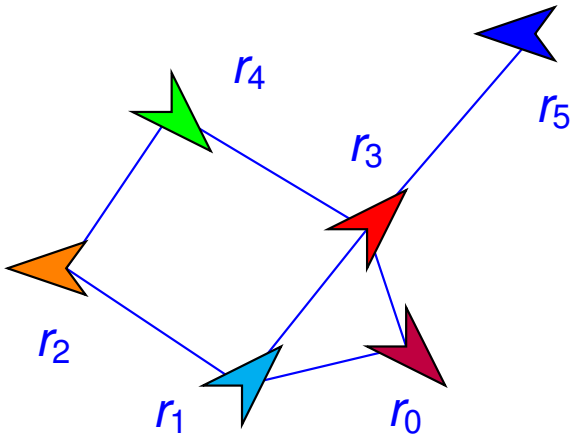


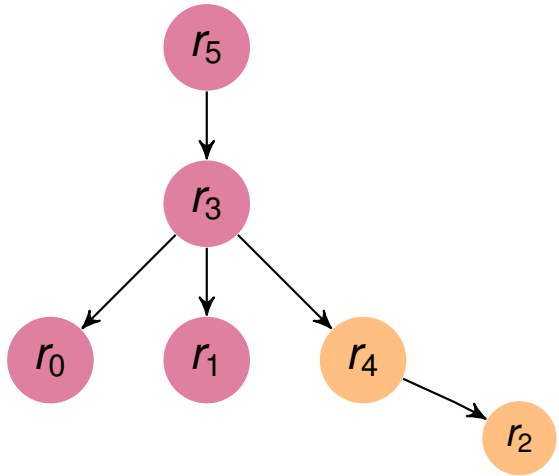
Spanning Tree Construction

During the communication process, each robot computes its message and constructs a spanning tree. We set a *Communication Timeout* for all robots to reach consensus.

Definition

The **communication timeout** is the number of time steps that the robot needs to wait before it is confident about the principal information from the incoming messages.



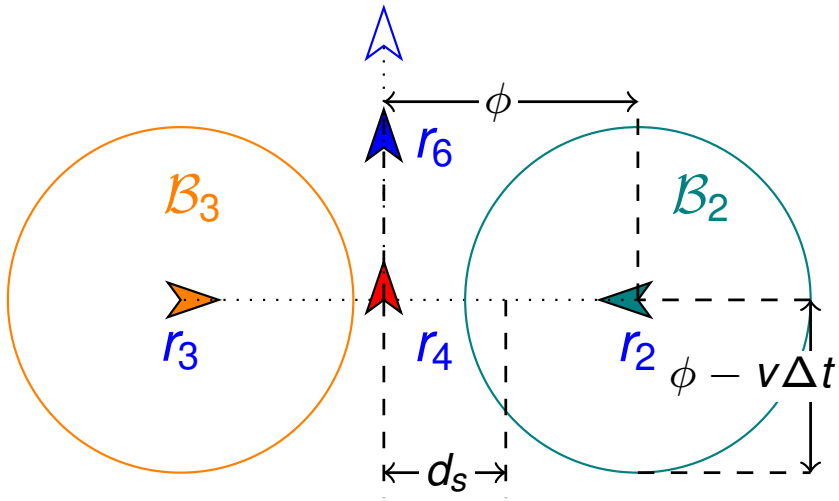


No Child Left Behind

Theorem

The motion strategy guarantees the connectivity of the communication graph.

- Only the relocate robot and its descendants (if any) move
- Moving robot only moves to where it guarantees to stay within the bounded range of each of its children.
- Non-relocate Moving robot moves close enough (safe distance d_s) to its parent

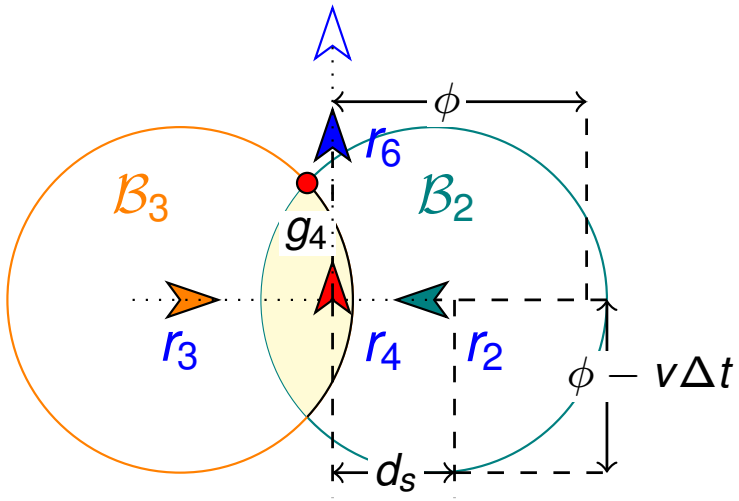


No Child Left Behind

Theorem

The motion strategy guarantees the connectivity of the communication graph.

- Only the relocate robot and its descendants (if any) move
- Moving robot only moves to where it guarantees to stay within the bounded range of each of its children.
- Non-relocate Moving robot moves close enough (safe distance d_s) to its parent



Correctness

Theorem

Given a set of robots, if the communication graph is initially connected, robots will reach positions satisfying the input lattice graph with bounded time.

Proof.

For the relocate robot r_i, the time it needs to reach a vacancy:

$$\begin{aligned} T_i &= T_{\text{timeout}} + T_{\text{wait}} + T_{\text{drive}} \\ &\leq O(n) + \phi(n-i-1)/v + i\phi/v \\ &\leq O(n) + (n-1)\phi/v \\ &= O(n). \end{aligned}$$

The total algorithm execution time for the robots to reach desired formation is $O(n^2)$.

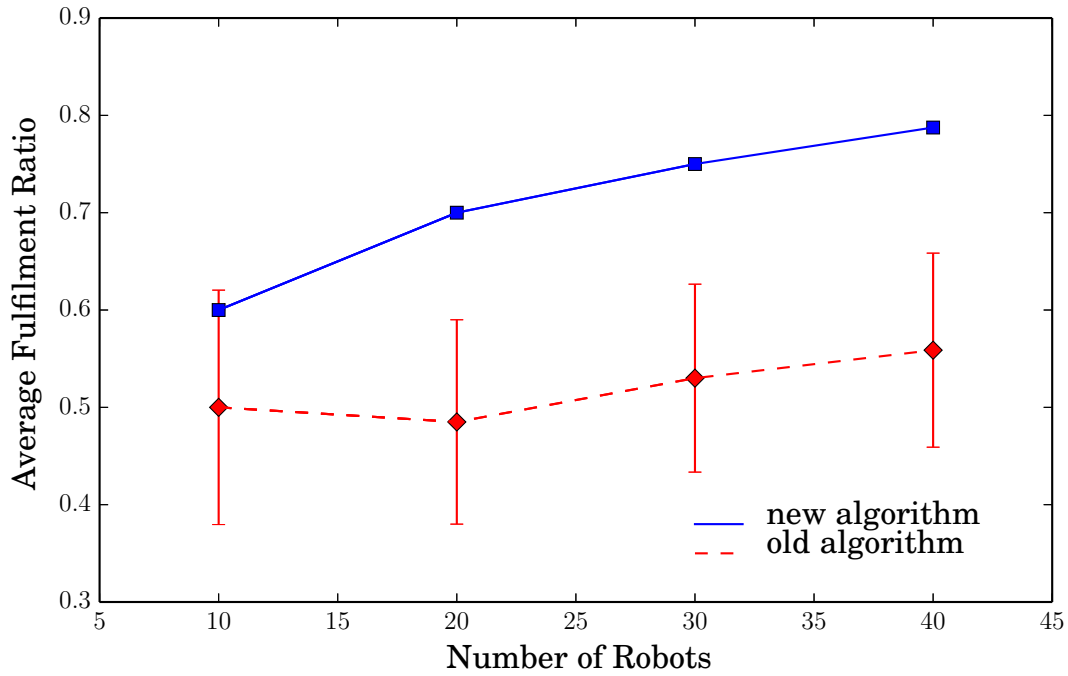
□

Experiments

Square Pattern

Goal: evaluate the execution time and formation quality

1. The number of robots $n = 10, 20, 30, 40$.
2. For each n, run 10 trials of simulations.
3. Random but connected initial communication graph.



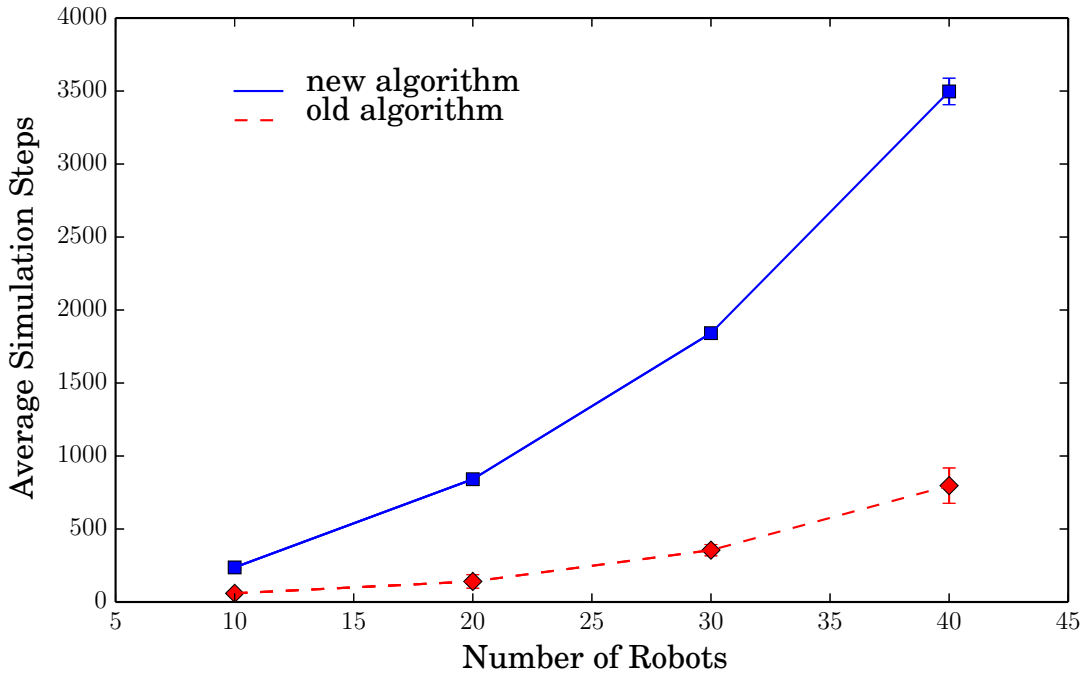
Simulation Quality Comparison

Experiments

Square Pattern

Goal: evaluate the execution time and formation quality

1. The number of robots $n = 10, 20, 30, 40$.
2. For each n, run 10 trials of simulations.
3. Random but connected initial communication graph.



Simulation Steps Comparison

Simulation: Repeated Hexagon

32 robots

Simulation: Repeated Octagon-Square

32 robots

Simulation: Finite Pattern

* Collaborative work with Asante Dawkins

Summary

1. Proved bonded execution time.
2. Deterministic formation quality, always better than old algorithm.

Discussions

1. Only one robot relocate to one vacancy.
2. Lack of error-tolerance.

Prior Work

- Probabilistic representations: J. van den Berg, P. Abbeel, and K. Goldberg (LQG-MP, IJRR 2011)
- Minimal Representations: B. Tovar, L. Guilamo, S. M. LaValle (GNT, WAFR 2004)
- Geometric over-approximation: W. Xu, J. M. O'Kane (Wireless Netw 2012)

Contributions

Contributions

- Define the range space $\mathcal{R}$ with formulation of the operations
- Design algorithms for double-rectangle range space $\mathcal{R}_{drect}$
- Conduct experiments to evaluate different range spaces

Example

Navigation Task using I-state

Contribution: Range Space

Definition

A range space $\mathcal{R} \subseteq \mathcal{I}$ is a set of I-states, equipped with two operations:

1. *Approximate action update function* $T : \mathcal{R} \times U \to \mathcal{R}$:

$$\bigcup_{x_k \in \eta_k} F(x_k, u_k) \subseteq T(A(\eta_k), u_k)$$

2. *Approximate observation update function* $O : \mathcal{R} \times Y \to \mathcal{R}$:

$$\eta_k \cap H(y_k) \subseteq O(A(\eta_k), u_k)$$

Intuition: always over-approximate
I-state: $\eta_k \subseteq A(\eta_k)$

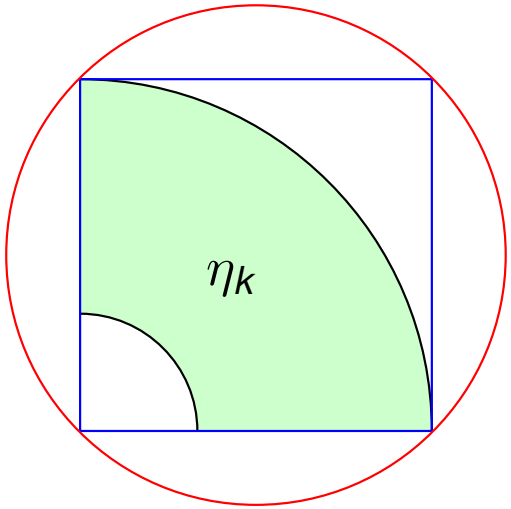


Figure: An *I-state* η_k (shaded region), over-approximations $A(\eta_k)$ using disk (red circle) and rectangle (blue).

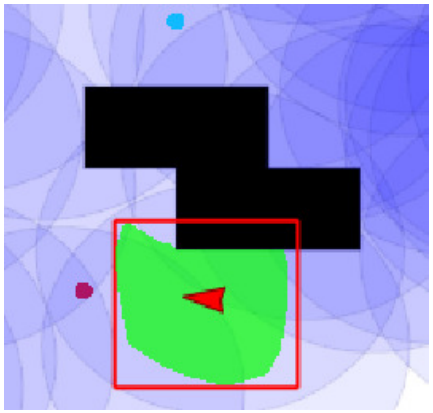
Example

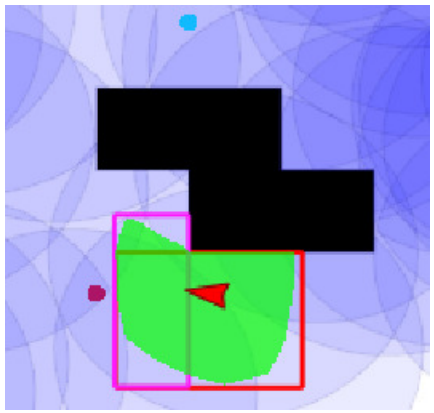
Rectangle Approximation of I-state

Limitation

Rectangle Approximation of I-state

Problem: approximation accuracy





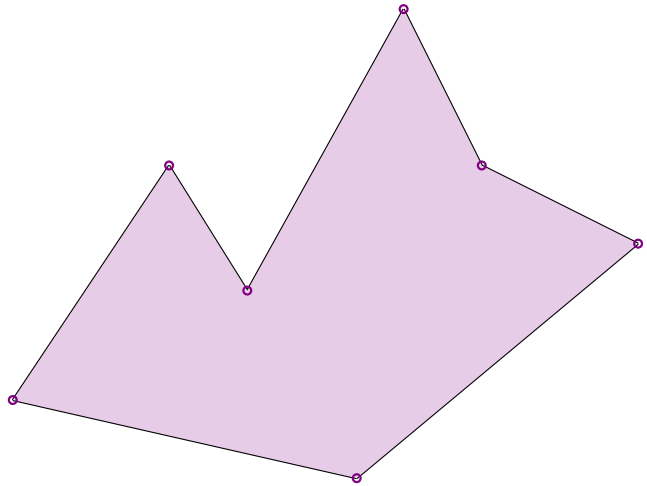
Double-Rectangle approximated I-state

To improve the approximation quality for non-convex *I-states*, we proposed:

- a novel range space of double-rectangle

$$\mathcal{R}_{drect} = \{R_1 \cup R_2 \mid R_1, R_2 \in \mathcal{R}_{rect}\} \tag{1}$$

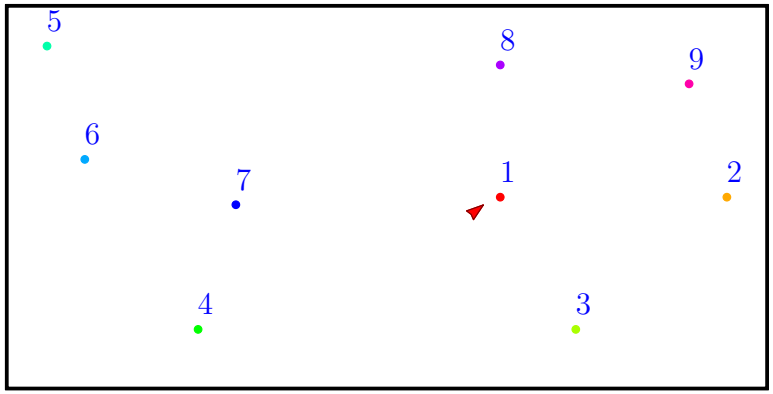
- an algorithm to overapproximate a polygon using double-rectangle

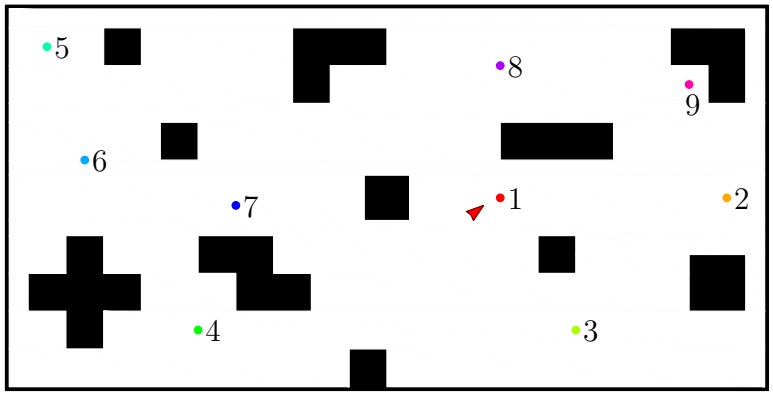


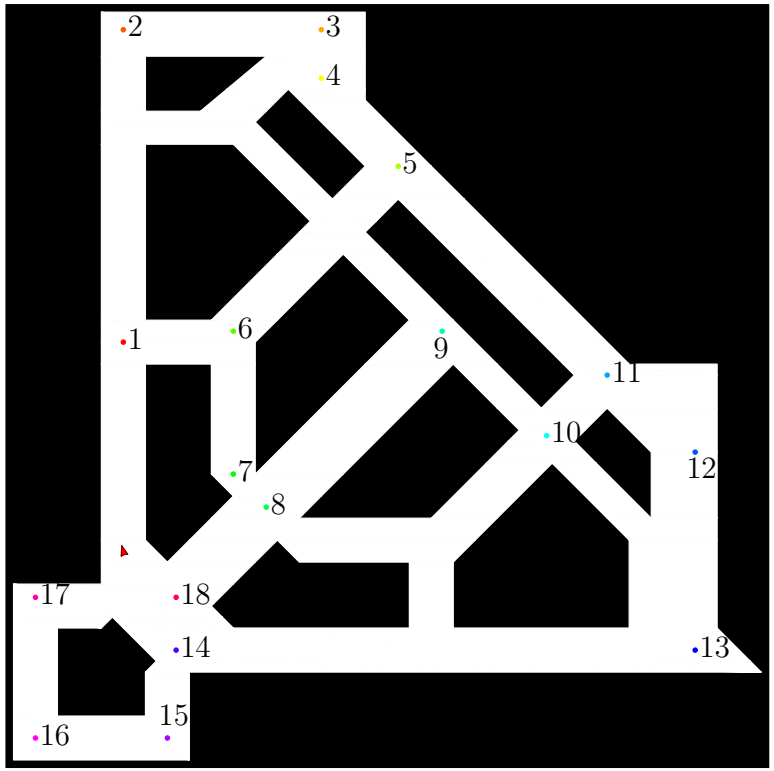
Simulation

Double-Rectangle Approximation of I-state

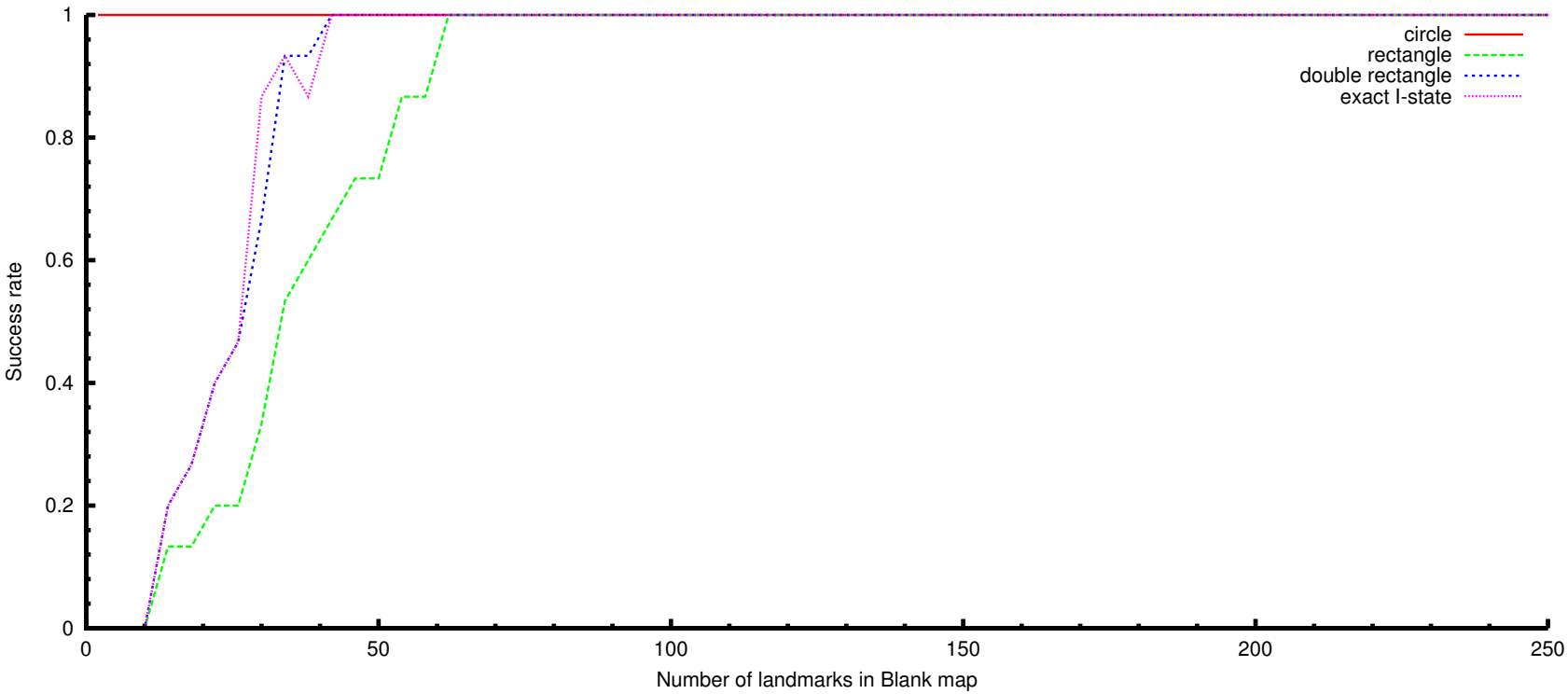
Experiments





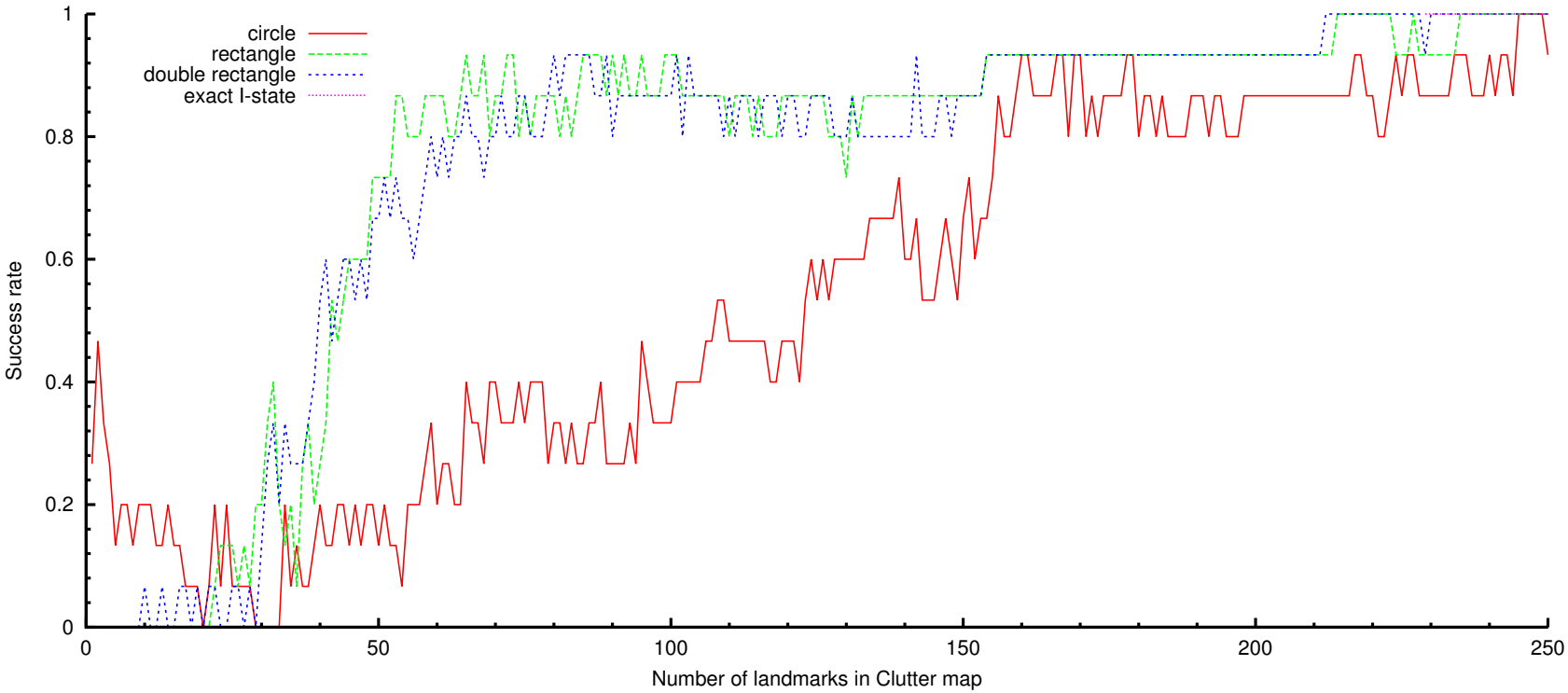


Experiments



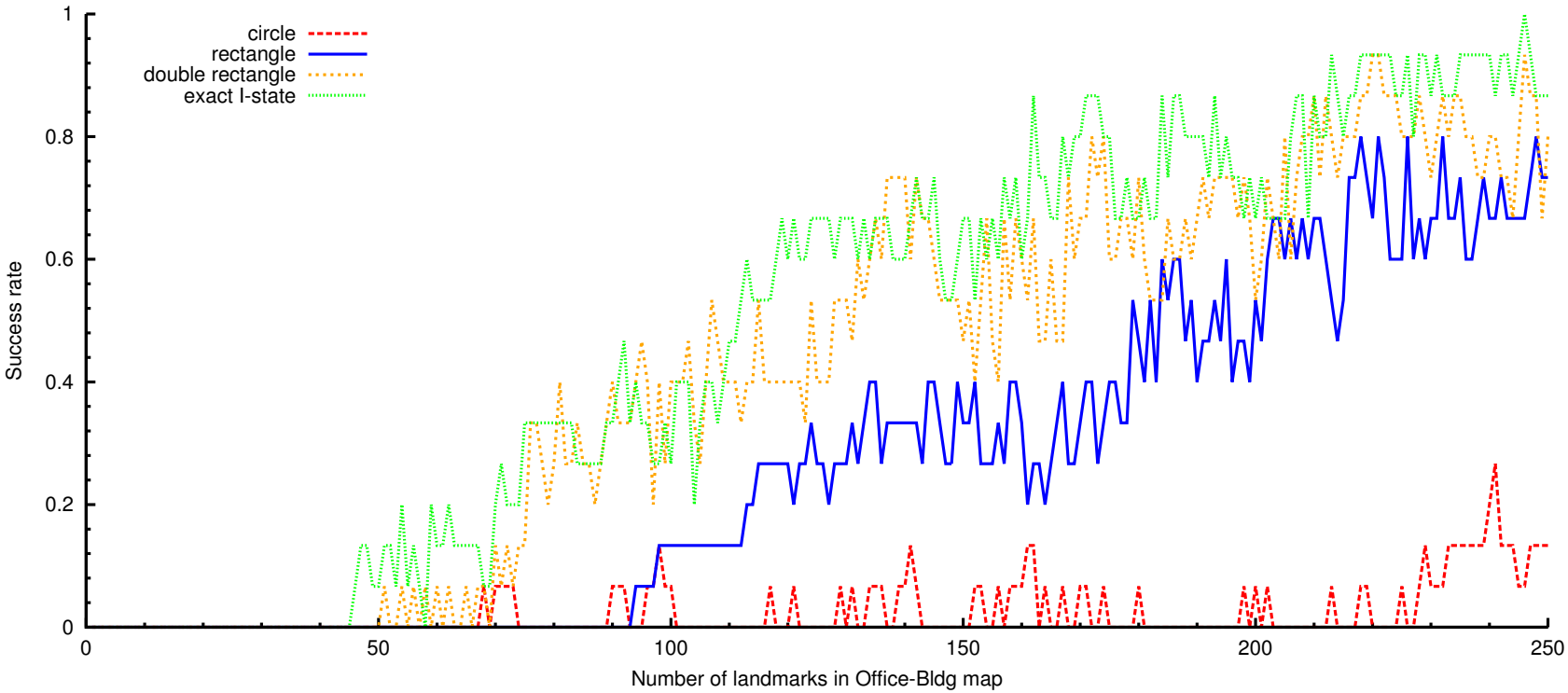
Success Rate for the Obstacle-free Environment

Experiments



Success Rate for the Obstacle-cluttered Environment

Experiments



Success Rate for the Office-like Environment

Experimental Results

Computation Time And Approximation Ratio

For each environment, we randomly place 300 landmarks and run 10 trials of simulations.
We measure:

- Average time (seconds) to compute the approximated I-state and the exact I-state.
- Average approximation ratio of using each approximation methods to approximate the exact I-state $\frac{1}{k}\sum_{i=1}^{k}\frac{\text{AREA}(\eta_i)}{\text{AREA}(A(\eta_i))}$.

Range Space	Computation Time (second)		
	Obstacle-free	Obstacle-cluttered	Office-like
$\mathcal{R}_{disk}$	0.163	0.162	0.292
$\mathcal{R}_{rect}$	0.396	0.441	0.415
$\mathcal{R}_{dblrect}$	1.021	1.122	1.491
$\mathcal{I}$	10.074	10.218	26.895

Table: Average computation time of approximated I-states and exact I-states and in three environments.

Experimental Results

Computation Time And Approximation Ratio

For each environment, we randomly place 300 landmarks and run 10 trials of simulations.
We measure:

- Average time (seconds) to compute the approximated I-state and the exact I-state.
- Average approximation ratio of using each approximation methods to approximate the exact I-state $\frac{1}{k}\sum_{i=1}^{k}\frac{\text{AREA}(\eta_i)}{\text{AREA}(A(\eta_i))}$.

Range Space	Approximation Ratio		
	Obstacle-free	Obstacle-cluttered	Office-like
$\mathcal{R}_{disk}$	0.155	0.155	0.220
$\mathcal{R}_{rect}$	0.642	0.632	0.661
$\mathcal{R}_{dblrect}$	0.684	0.691	0.720
$\mathcal{I}$	1.000	1.000	1.000

Table: Average approximation ratio in three environments.

Conclusions

1. CGA is effective for representing a robot's uncertain information about the current state
2. The form of double-rectangle is more accurate in approximating the non-convex I-state
3. The robot can complete the navigation task using approximated I-state with low approximation accuracy

Conclusions

Contributions:

1. Geometric data structure to over-approximate I-state.
2. Graph representations of diverse lattice patterns.
3. Distributed task-assignment-based formation algorithm.
4. “No Child Left Behind” movement strategy.
5. Bounded time performance formation algorithm.

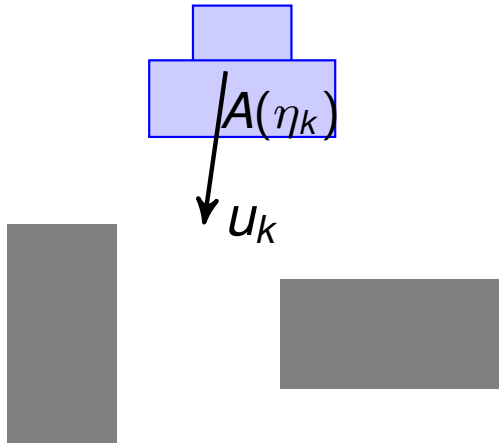
Future Work:

- k-fold unions of rectangles for approximation.
- k spanning trees with k vacancies for formation.

Operations on Double-Rectangle Range Space

For a double rectangle approximated *I-state* $A(\eta_k) = R_1 \cup R_2$:

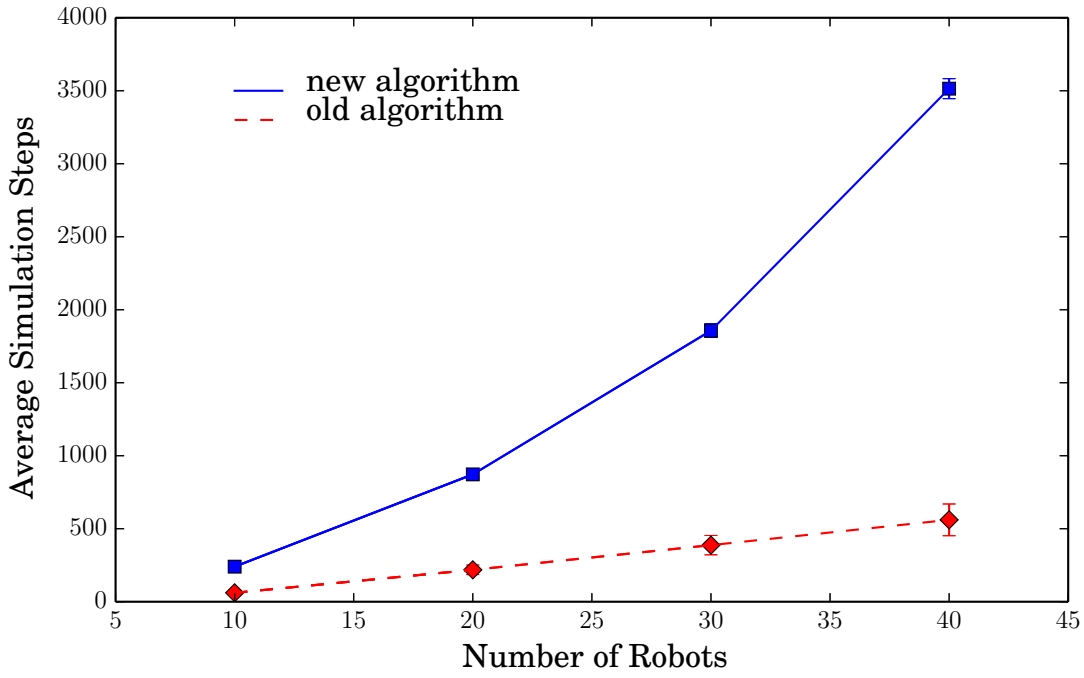
- Action Update:
  $T_{drect}(A(\eta_k), u_k) = DRAP(X_{free} \cap [A(\eta_k) \oplus \{u_k\} \oplus DRAP(\Theta(u_k))])$
- Observation Update:
  $O_{drect}(A(\eta_k), y_k) = AABB(H(y_k) \cap R_1) \cup AABB(H(y_k) \cap R_2)$



Experiments

Hexagon pattern

1. The number of robots $n = 10, 20, 30, 40$.
2. For each n, run 10 trials of simulations.
3. Connected initial communication graph.

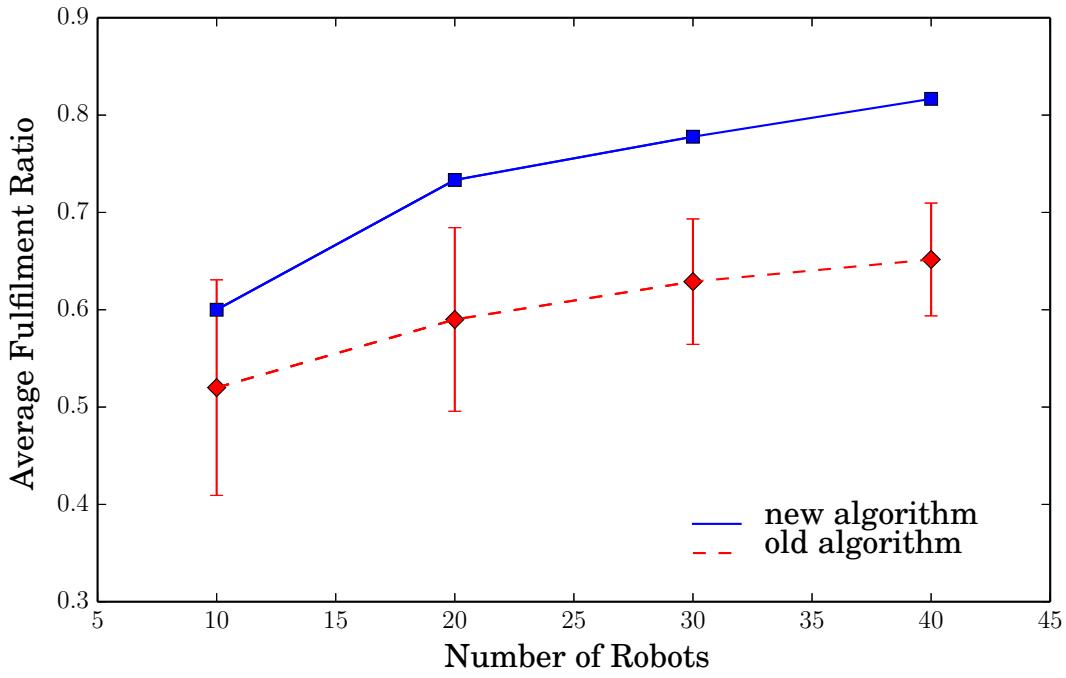


Simulation Steps Comparison

Experiments

Hexagon pattern

1. The number of robots $n = 10, 20, 30, 40$.
2. For each n, run 10 trials of simulations.
3. Connected initial communication graph.

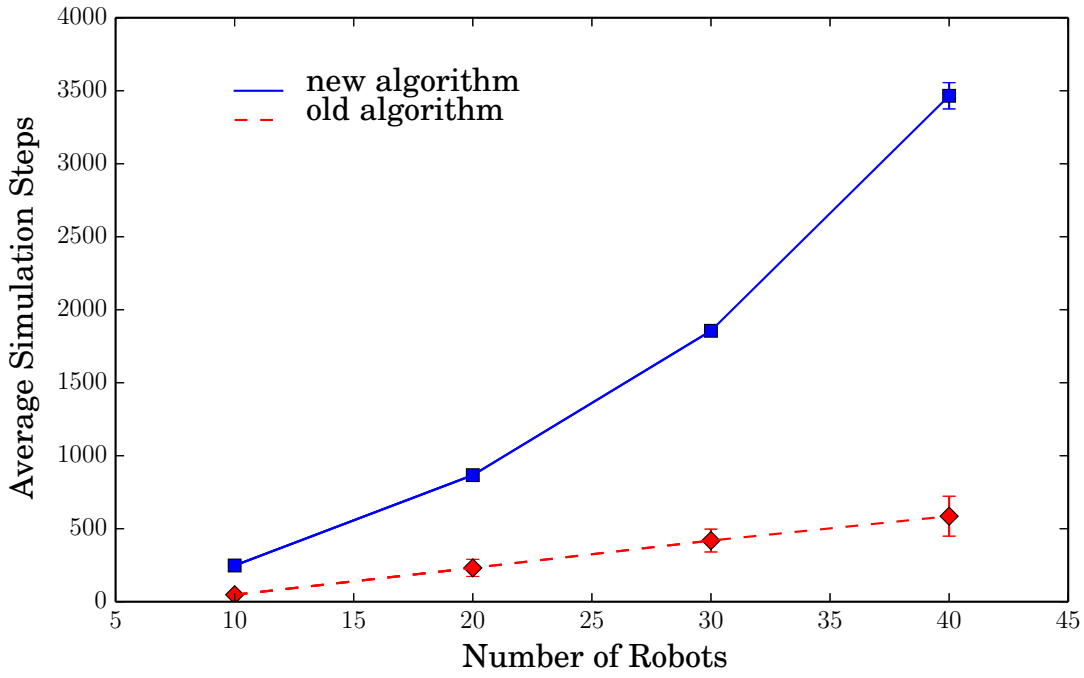


Simulation Quality Comparison

Experiments

Octagon-Square Pattern

1. The number of robots $n = 10, 20, 30, 40$.
2. For each n, run 10 trials of simulations.
3. Connected initial communication graph.

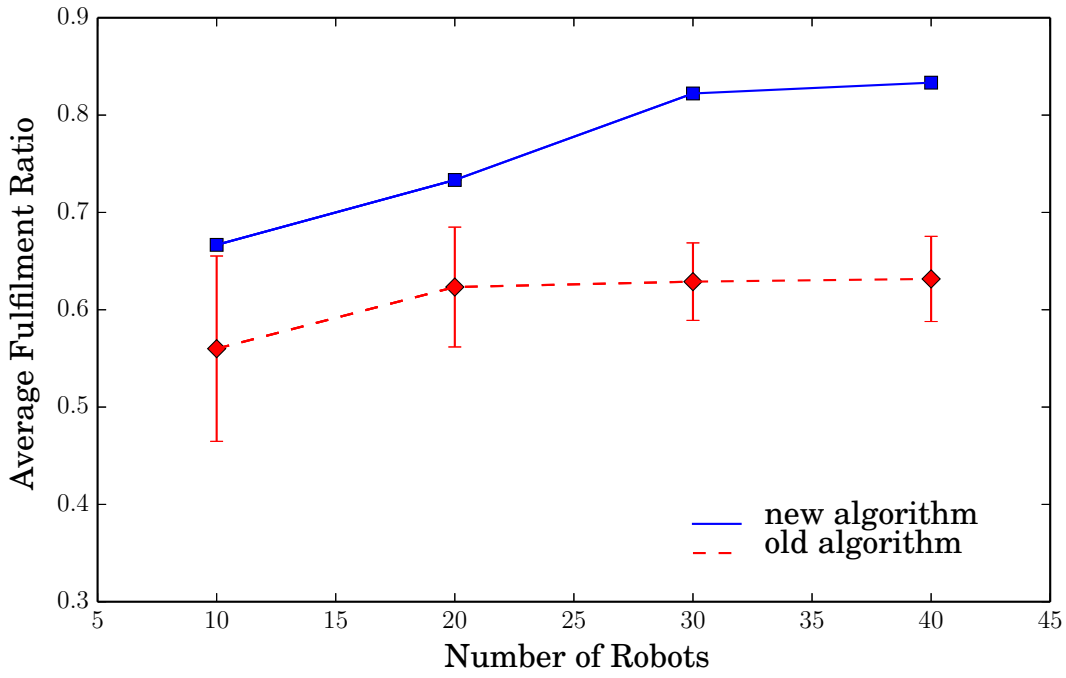


Simulation Steps Comparison

Experiments

Octagon-Square Pattern

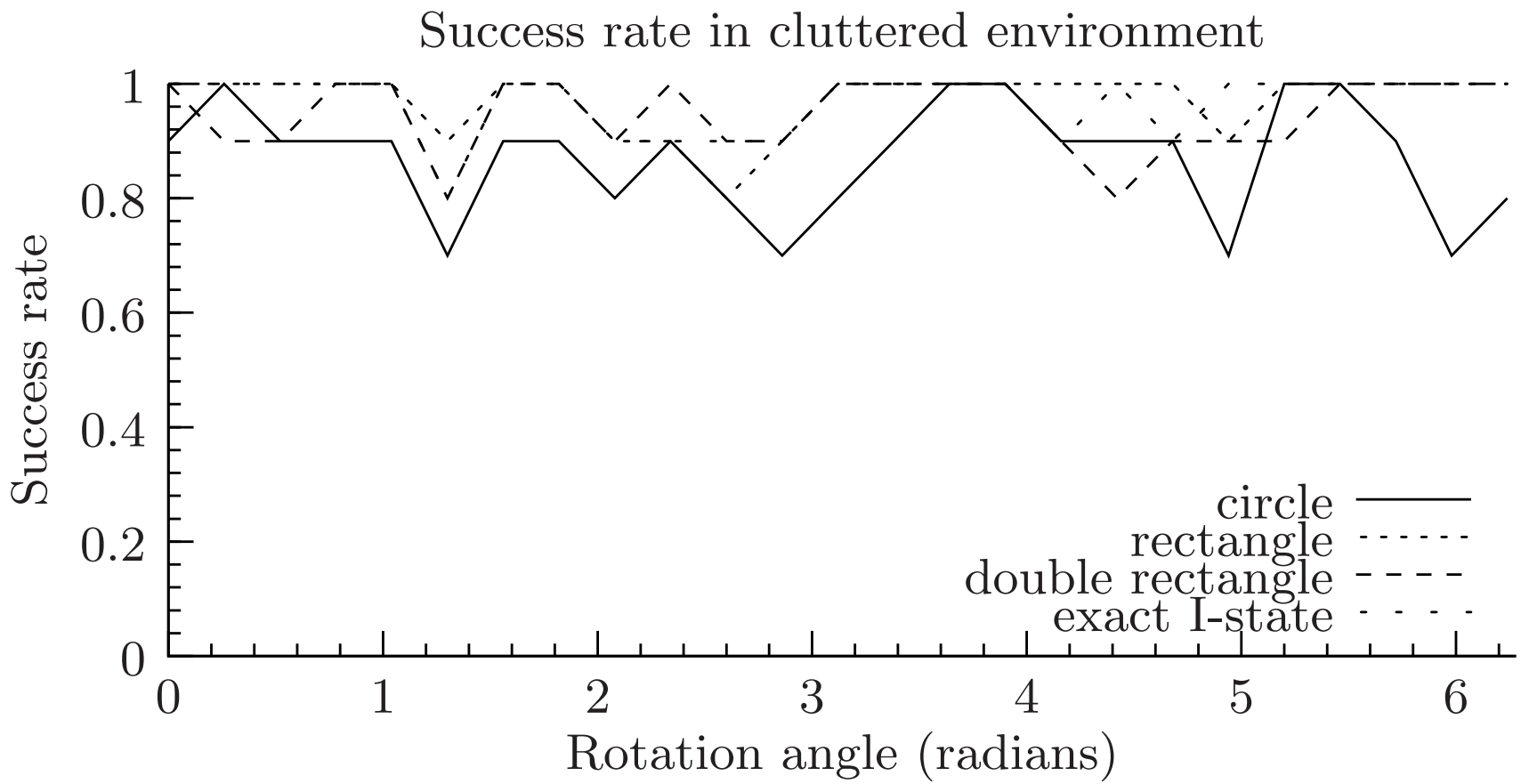
1. The number of robots $n = 10, 20, 30, 40$.
2. For each n, run 10 trials of simulations.
3. Connected initial communication graph.



Simulation Quality Comparison

Experimental Results

For Rotated Environments



Experimental Results

For Rotated Environments