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

Yang Song, Jason M. O'Kane

song24@email.sc.edu

jokane@cse.sc.edu

Dept. of Computer Science and Engineering
University of South Carolina

UNIVERSITY OF SOUTH CAROLINA

Outlines

Constrained Geometric Approximation

- **Goal**:
  For an extremely simple robot with:
  - computation limitations
  - moving and sensing uncertainties

  represent and reason about uncertainty in its own states efficiently.
- **Basic Idea**:
  Explicitly represent what the robot knows as an information state (*I-state*).
- **Intuition**:
  Accelerate time-consuming operations by maintaining only an overapproximation of the true *I-state*, and constraining this approximation to have a simple geometric form.

SRV-1 Surveyor Robot

Robot Model

Assume that current real state of the robot could not be observed directly. The robot could maintain an *I-state* η_k, to make its decisions.

- Robot state at stage k: $x_k \in X$,
- State transition function: $F(x_k, u_k)$.
- Robot action at stage k: u_k.
- Information state (*I-state*) at stage k: η_k is a set of possible states at stage k

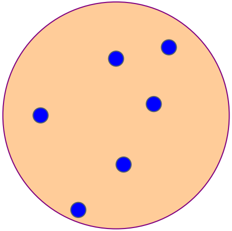


Figure: I-state η_k contains all possible states

Example Application
Rectangle Approximation

Prior Work

- Prior research done by (B. Tovar and S. M. LaValle) and (J. van den Berg, P. Abbeel, and K. Goldberg) used probabilistic representations for planning
- Prior work by the J.O'Kane has used preliminary versions of the constrained geometric approximation method using specific, fixed range spaces.

New contributions

1. A careful formulation of the operations in the range space $\mathcal{R}$.
2. Algorithms for double-rectangle range space $\mathcal{R}_{direct}$.
3. A series of experiments for effectiveness comparison of different range spaces.

Outlines

Range Space

Definition

A range space $\mathcal{R} \subseteq \mathcal{I}$ is a set of I-states, contains approximation of *I-states*, $A(\eta_k) \in \mathcal{R}$, equipped with two operations:

1. An *approximate observation update function* $O : \mathcal{R} \times Y \to \mathcal{R}$, such that if $\eta_k \subseteq A(\eta_k)$, then

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

2. An *approximate action update function* $T : \mathcal{R} \times U \to \mathcal{R}$, such that if $\eta_k \subseteq A(\eta_k)$, then

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

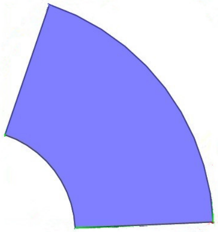


Figure: An *I-state* η_k

Definition

A range space $\mathcal{R} \subseteq \mathcal{I}$ is a set of I-states, contains approximation of *I-states*, $A(\eta_k) \in \mathcal{R}$, equipped with two operations:

1. An *approximate observation update function* $O : \mathcal{R} \times Y \to \mathcal{R}$, such that if $\eta_k \subseteq A(\eta_k)$, then

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

2. An *approximate action update function* $T : \mathcal{R} \times U \to \mathcal{R}$, such that if $\eta_k \subseteq A(\eta_k)$, then

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

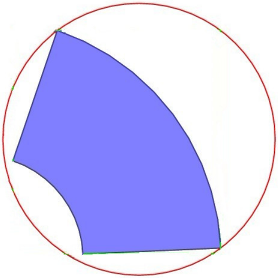


Figure: Disk overapproximation $A(\eta_k) \in \mathcal{R}_{disk}$ in red.

Range Space

Definition

A range space $\mathcal{R} \subseteq \mathcal{I}$ is a set of I-states, contains approximation of *I-states*, $A(\eta_k) \in \mathcal{R}$, equipped with two operations:

1. An *approximate observation update function* $O : \mathcal{R} \times Y \to \mathcal{R}$, such that if $\eta_k \subseteq A(\eta_k)$, then

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

2. An *approximate action update function* $T : \mathcal{R} \times U \to \mathcal{R}$, such that if $\eta_k \subseteq A(\eta_k)$, then

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

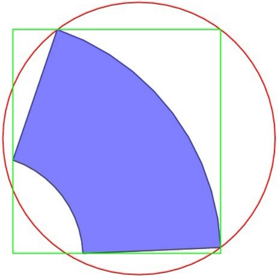
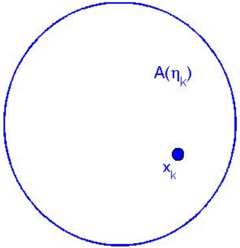


Figure: Rectangle overapproximation $A(\eta_k) \in \mathcal{R}_{rect}$.

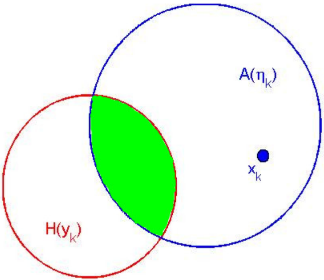
Observation Update in $\mathcal{R}_{disk}$

Approximated I-state $A(\eta_k) \in \mathcal{R}_{disk}$, where x_k denotes the real state but unknown to robot.



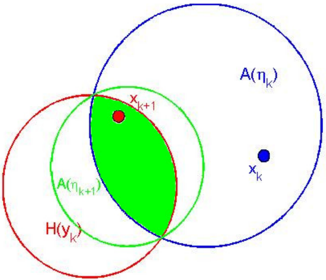
Observation Update in $\mathcal{R}_{disk}$

Approximated I-state $A(\eta_k) \in \mathcal{R}_{disk}$ intersects with observation preimage $H(y_k)$.



Observation Update in $\mathcal{R}_{disk}$

The green region of intersection is the updated I-state η_{k+1}, and the green disk is the approximation $A(\eta_{k+1})$ of η_{k+1}.



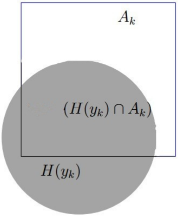
Observation Update in $\mathcal{R}_{rect}$

Definition

$AABB(S)$: For any compact set $S \subset \mathbb{R}^2$, let $AABB(S)$ denote the its smallest "axis-aligned bounding box."

In $\mathcal{R}_{rect}$, computing approximate observation update function O_{rect} takes $O(1)$ time:

$$O_{rect}(A_k, y_k) = AABB(H(y_k) \cap A(\eta_k)), A_k = A(\eta_k) \in \mathcal{R}_{rect}$$



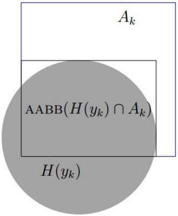
Observation Update in $\mathcal{R}_{rect}$

Definition

$AABB(S)$: For any compact set $S \subset \mathbb{R}^2$, let $AABB(S)$ denote the its smallest "axis-aligned bounding box."

In $\mathcal{R}_{rect}$, computing approximate observation update function O_{rect} takes $O(1)$ time:

$$O_{rect}(A_k, y_k) = AABB(H(y_k) \cap A(\eta_k)), A_k = A(\eta_k) \in \mathcal{R}_{rect}$$



Action Update in $\mathcal{R}_{rect}$

In $\mathcal{R}_{rect}$, computing approximate action update function T_{rect} :

$$T_{rect}(A(\eta_k), u_k) = AABB(X_{free} \cap [A(\eta_k) \oplus \{u_k\} \oplus AABB(\Theta(u_k))])$$

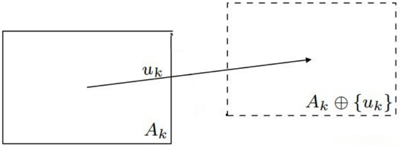


Figure: Transition of $A(\eta_k)$ given action u_k, $\oplus$ is Minkowski addition of two sets.

Action Update in $\mathcal{R}_{rect}$

In $\mathcal{R}_{rect}$, computing approximate action update function T_{rect} :

$$T_{rect}(A(\eta_k), u_k) = AABB(X_{free} \cap [A(\eta_k) \oplus \{u_k\} \oplus AABB(\Theta(u_k))])$$

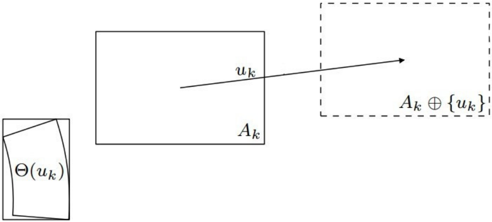


Figure: Consider approximation of the bounded motion uncertainty $\Theta(u_k)$.

Action Update in $\mathcal{R}_{rect}$

In $\mathcal{R}_{rect}$, computing approximate action update function T_{rect} :

$$T_{rect}(A(\eta_k), u_k) = AABB(X_{free} \cap [A(\eta_k) \oplus \{u_k\} \oplus AABB(\Theta(u_k))])$$

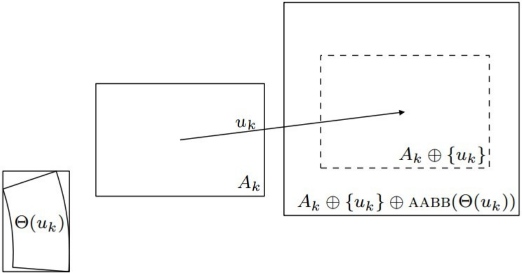


Figure: Find Minkowski sum of bounded noise and approximation transition

Action Update in $\mathcal{R}_{rect}$

In $\mathcal{R}_{rect}$, computing approximate action update function T_{rect} :

$$T_{rect}(A(\eta_k), u_k) = AABB(X_{free} \cap [A(\eta_k) \oplus \{u_k\} \oplus AABB(\Theta(u_k))])$$

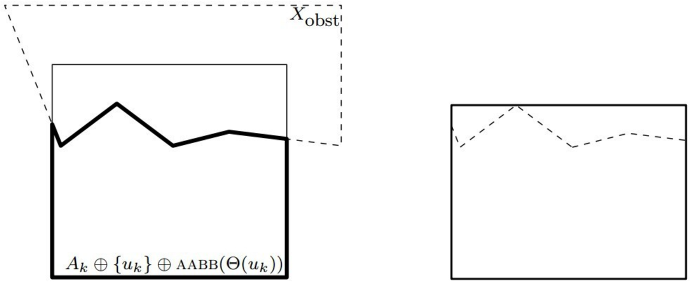


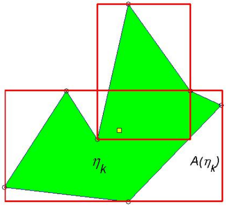
Figure: If there is obstacles, intersect with X_{free} first and then find the bounding box of the intersection.

Double-Rectangle approximated I-state

- For better overapproximation quality for non-convex *I-states*, we proposed a more expressive range space of *double rectangles*:

$$\mathcal{R}_{direct} = \{R_1 \cup R_2 \mid R_1, R_2 \in \mathcal{R}_{rect}\} \qquad (1)$$

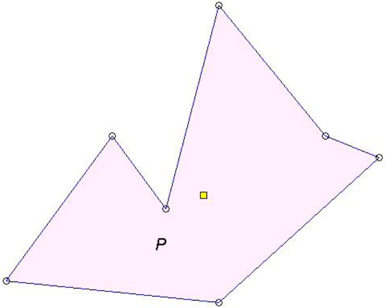
- Aims to improve the approximation quality.



DRAP Algorithm

"Double Rectangle Around Polygon" (**DRAP**) algorithm:

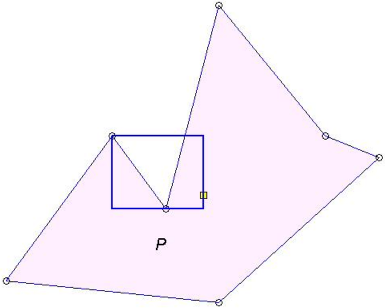
- input is a n-edge polygonal region of the plane
- output is a small double rectangle containing that polygon
- run time: $O(n^3)$



DRAP Algorithm

"Double Rectangle Around Polygon" (**DRAP**) algorithm:

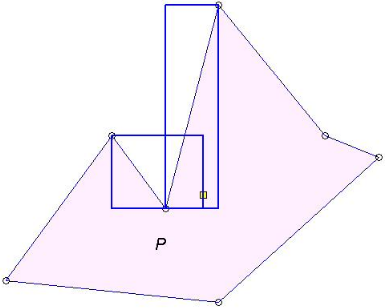
- input is a n-edge polygonal region of the plane
- output is a small double rectangle containing that polygon
- run time: $O(n^3)$



DRAP Algorithm

"Double Rectangle Around Polygon" (**DRAP**) algorithm:

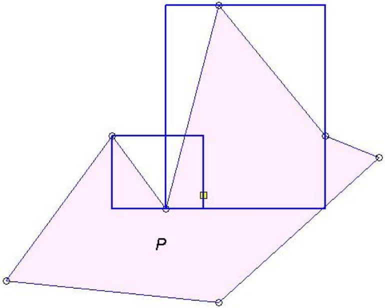
- input is a n-edge polygonal region of the plane
- output is a small double rectangle containing that polygon
- run time: $O(n^3)$



DRAP Algorithm

"Double Rectangle Around Polygon" (**DRAP**) algorithm:

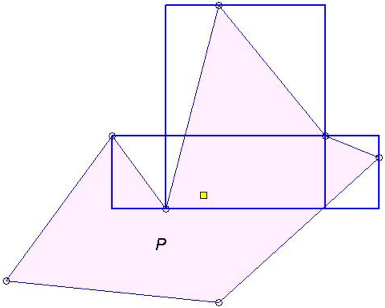
- input is a n-edge polygonal region of the plane
- output is a small double rectangle containing that polygon
- run time: $O(n^3)$



DRAP Algorithm

"Double Rectangle Around Polygon" (**DRAP**) algorithm:

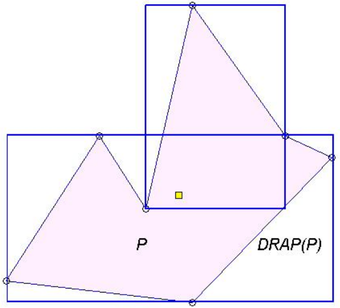
- input is a n-edge polygonal region of the plane
- output is a small double rectangle containing that polygon
- run time: $O(n^3)$



DRAP Algorithm

"Double Rectangle Around Polygon" (**DRAP**) algorithm:

- input is a n-edge polygonal region of the plane
- output is a small double rectangle containing that polygon
- run time: $O(n^3)$



Outlines

CGA for Robot Planning

Range Space
- Disk Range Space
- Rectangle Range Space
- Double-Rectangle Range Space

Experiments and Conclusions

Multi-Robot Formation

Lattice Graph
- Algorithm
- Robot Authority

Experiments in Various Environments

Comparison with using the true I-state, we conducted experiments using 3 environments, and 3 range spaces $\mathcal{R}_{disk}$, $\mathcal{R}_{rect}$, and $\mathcal{R}_{drect}$.

ASSUMPTIONS:

- Robot is guided by centroid point of the approximated *I-state*
- Robot can detect presence but not distance to the settled waypoints
- Landmarks are pseudo-randomly generated (Number Matters)
- Initial I-state η_0 is given.

Experiments Results

For each environment, we measure:

- the relationship between task completion and number of landmarks,
- the time required to compute approximated *I-state* compared to the high-quality polygonal representation of the exact *I-state*,
- the approximation ratio Q_k

$$Q_k = \frac{1}{k} \sum_{i=1}^{k} \frac{\mathbb{A}(\eta_i)}{\mathbb{A}(A(\eta_i))} \tag{2}$$

in which $\mathbb{A}(\diamond)$ denotes the area of set $\diamond \subset \mathbb{R}^2$

Comparison of Efficiency and Accuracy

Range Space	Obstacle-free Time(s)	Obstacle-free Approx.Ratio	Clutter Time(s)	Clutter Approx.Ratio	Office-like Time(s)	Office-like Approx.Ratio
$\mathcal{R}_{disk}$	0.163	0.155	0.162	0.155	0.292	0.220
$\mathcal{R}_{rect}$	0.396	0.642	0.441	0.632	0.415	0.661
$\mathcal{R}_{dblrect}$	1.021	0.684	1.122	0.691	1.491	0.720
$\mathcal{I}$	10.074	1.000	10.218	1.000	26.895	1.000

Table: Experimental results: computation time and approximation ratio for range spaces and I-space in three test environments.

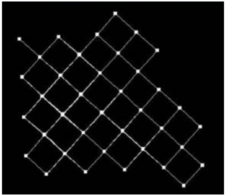
Conclusions

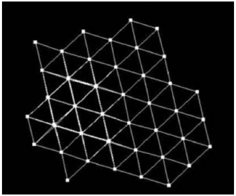
- CGA is effective for representing a robot's uncertain information about the current state.
- The form of double-rectangle is more accurate in approximating the non-convex *I-state*.
- The robot can complete the navigation task using approximated I-state with low approximation accuracy.

Related Work

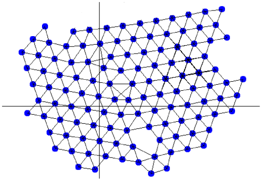
Formation using virtual force

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

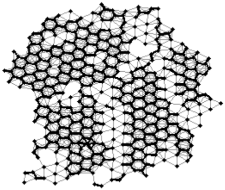




I. Navarro, J. Pugh, A. Martinoli, and F. Matia, 2008

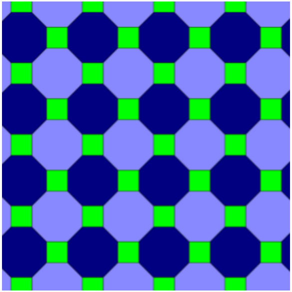


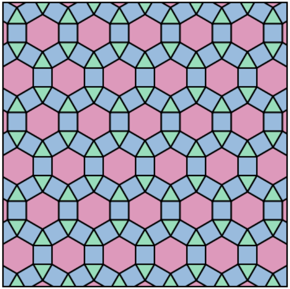
S. Prabhu, W. Li, J. McLurkin, 2012



Motivation

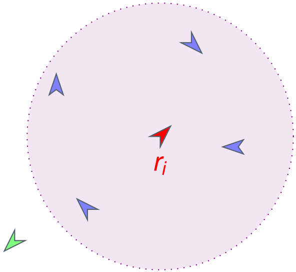
Question: How to use one algorithm to generate various (repeating) lattice pattern formations?





Robot Model

- Differential Drive robots.
- Each robot has an unique **ID**.
- Use a vector $p = [x, y, \theta]^T$ to represent robot's **pose**.
- 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.

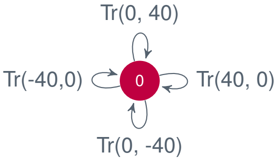


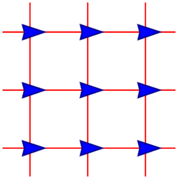
Robot r_i has 4 neighbors

Input: Lattice Graph

Definition

A **lattice graph** is a strongly connected directed multigraph 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 $w \xrightarrow{T(e)^{-1}} v$.





Simulation Example

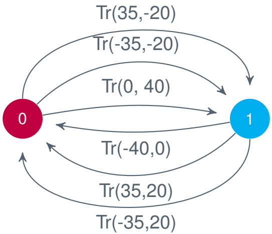
Square formation with 10 robots

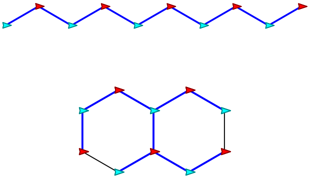
Role Function

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 $f : R \rightarrow V$ that preserves the neighborhood structure of G.

Specifically, for any i and j, if r_i and r_j are neighbors, there must exist an edge $e_{ij} : f(r_i) \longrightarrow f(r_j)$ in E, such that $T(r_j) = T(r_i)T(e_{ij})$.





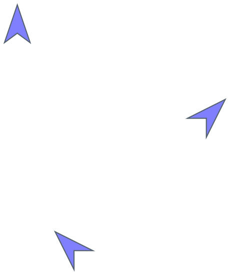
Algorithm

General Description

Robot broadcasts message containing its

- authority
- matching.

1. Form tree structure.
2. Use tree structure to computer local task assignment.
3. Make movement decision.



Authority

Define authority and comparison operator

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.

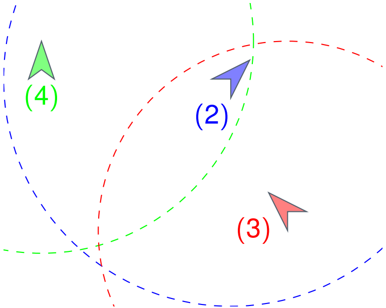
1.Construct Authority Tree

Decide to be root or descendant

The robots use these authorities to establish a collection of authority trees

1. Discards any message in which the authority contains its own ID.
2. Forms an authority containing only its own ID, compares it with the authorities of remaining messages and selects the highest authority.

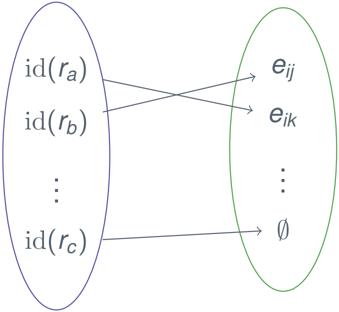
- If its authority is the highest, then it is a root;
- Otherwise, it selects the one who sends the highest authority as its parent. Append its own ID to the highest authority to create its own authority.



Matching

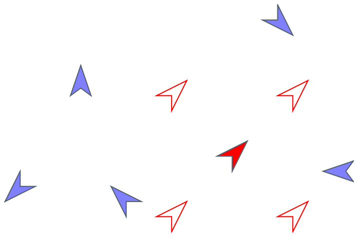
Definition

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



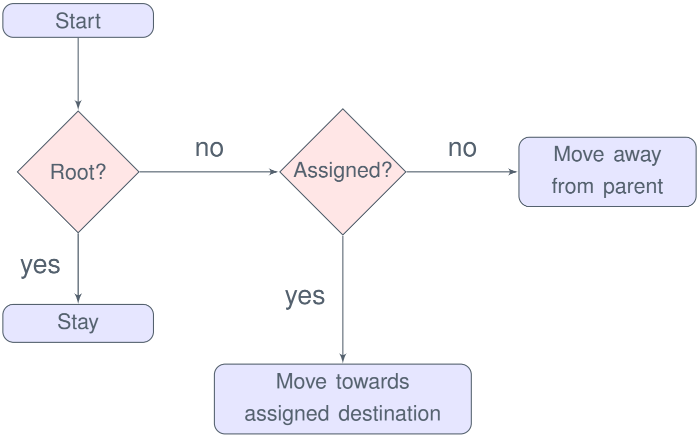
To compute an optimal matching of a robot with N neighbors and E out-going edges of its role in the lattice graph, define a weight matrix of size $\min(N, E) \times E$ and apply Hungarian Algorithm (Harold W. Kuhn, 1955).

1. Each row corresponds to a neighbor;
2. Each column corresponds to an out-going edge of robot's role or a null value $\emptyset$.
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



Simulations

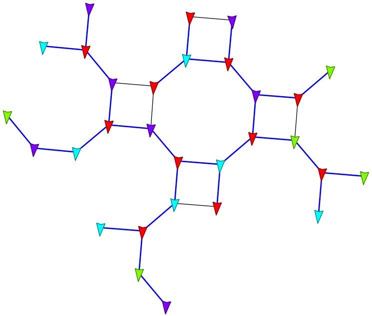


Figure: Octagon-square pattern formed by 30 robots.

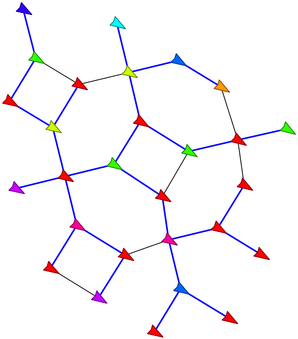


Figure: Hexagon-square pattern formed by 30 robots.