# A Provably-Correct Distributed Arbitrary Lattice Formation Algorithm for Large-Scale Multi-Robot Systems

Yang Song and Jason M. O'Kane

***Abstract*— We describe a new distributed algorithm for multi-robot systems to form arbitrary lattice patterns. In our prior work, we showed how to represent the desired lattice pattern using a directed graph in which each edge is labeled a rigid body transformation, and proposed an algorithm that accepts this graph as input and computes destinations for each robot using only local information. In this paper, we resolve several limitations of that algorithm by introducing changes to the information exchange protocol, motion strategy, and task assignment procedure. We prove that, by executing this improved algorithm, the robots will correctly form the desired lattice pattern in a bounded amount of time. Using a simulation, we demonstrate that this algorithm works correctly for systems with dozens of autonomous robots to form various lattice patterns. Moreover, the experiments show better formation quality can, in fact, be reached when using our new algorithm compared to our previous approach.**

## I. Introduction

Multi-robot systems have attracted a growing number of research efforts in recent years. In particular, the problem of creating formations of multiple robots been considered in multiple contexts, including autonomous ground, aerial, and underwater vehicles. Existing methods use either centralized or distributed algorithms to solve certain formation problems, subject to different constraints such as robot models, communication limits, *etc.*. In our previous work, we proposed an arbitrary lattice pattern formation problem for large-scale multi-robot systems, along with a distributed algorithm in which each robot uses only local information that solves this problem under some circumstances [1].

The input to our previous algorithm is a representation of desired formation in the form of a directed graph in which each vertex represents a "role" for a robot to play in the formation, and each outgoing edge is labeled with rigid body transformation indicating the desired location of one of that robot's neighbors. As the algorithm proceeds, each robot continually broadcasts messages to the other nearby robots. Based on these messages, the robots organize themselves into a rooted tree structure called an authority tree, and then move to the positions assigned to them by their parents in this tree.

Although this algorithm performed reasonably well in our simulated experiments, it has two critical limitations. First, there is no guarantee that the robots will eventually form the desired formation and terminate the algorithm. Instead, the robots may oscillate under some conditions. Second, the motion strategy employed for "orphan" robots—those to whom the chosen parent cannot assign a role—was simply to move directly away from that parent. This motion strategy may cause the formation become disconnected, decreasing the overall solution quality.

Yang Song (song24@email.sc.edu) and Jason M. O'Kane (jokane@cse.sc.edu) are with the Department of Computer Science and Engineering, University of South Carolina, Columbia.

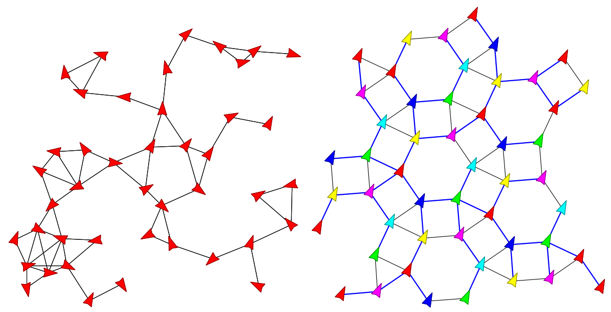


Fig. 1: [left] The initial poses of 50 robots, randomly generated with a uniform distribution. Line segments indicate communication connectivity between two robots. [right] The final formation after executing our algorithm using the triangle-hexagonal-square lattice graph. The robot colors indicate the roles selected by each robot.

In this paper, we introduce a new algorithm that resolves these limitations. Figure 1 shows an example of a repeating lattice pattern formed by 50 robots using our new algorithm. Specifically, this paper makes three new contributions beyond our prior work.

1) We describe a new algorithm which, while structurally similar to our existing work, differs in two essential ways. First, we define a new procedure to construct the authority tree, focusing on the combinatorial relationship of robots rather than their geometric positions. This change helps to prevent oscillations. Second, we introduce a new, more principled mechanism for controlling robots left as orphans in this tree structure. Robots using our new algorithm include information about the nearest opening in the partially formed lattice in each message they transmit. Orphan robots use this information to move toward those openings while remaining within communication range of the other robots. This change improves the quality of the final solution.
2) We prove that robots executing this new algorithm will form the desired lattice with bounded execution time. We also show that, if the communication graph is initially connected, then it will remain connected throughout the execution of the algorithm.
3) We empirically demonstrate the effectiveness of the new algorithm with a series of simulations.

In the following sections, first we give a short summary of related works on decentralized formation algorithms and control approaches in Section II. Then in Section III we describe the formation problem we are solving. We first recall the original algorithm upon which we improve in Section IV, then explain our new algorithm in detail in section V. Section VI proves that our algorithm terminates in finite time, with the robots at positions that satisfy the given lattice graph. We demonstrate simulated experiments to verify our algorithm and evaluate its performance in Section VII. In the final section, we conclude the paper.

## II. Related Work

Most works discussing the multi-robot lattice formation problem use distributed methods. Approaches using artificial virtual forces, such as the physicomimetics framework (PF), were popular in forming repeated lattice patterns, such as triangular, square, and hexagonal lattices [2], [3], [4], [5], [6], [7], [8]. Robots can quickly form specific lattice patterns using attractive or repulsive forces to adjust their positions based on only measurements of range and bearing to nearby robots. W. Spears, D. Spears, Heil, Zarzhitsky, and Hamann discussed swarm intelligence and applied PF-based methods to systems with multiple particle robots to form hexagonal lattices [3]. Martinson and Payton extended PF-based methods by first aligning robots to parallel lines, then applying virtual forces between robots to form square lattices [4]. Navarro, Pugh, Martinoli, and Matía implemented a PF-based algorithm as a finite state machine, so that the triangular lattice can be formed by the robots without assumption of holonomocity [5]. Based on the PF-based approach that is used for the triangular formation, Mullen, Monekosso, Barman, and Remagnino designed a cooperative formation control strategy to form hexagonal lattices. Their robots coordinate to determine the centroids of hexagonal cells and place virtual robot nodes in those positions to avoid local minima [8]. Prabhu, Li, and McLurkin designed a local error correction algorithm to detect and correct the misplacement in the hexagonal lattice formation using artificial forces [7].

However, a common limitation of PF-based algorithms is that each method is used for exclusively one specific lattice pattern formation. In contrast, Chaimowicz, Michael, and Kumar addressed a decentralized control framework to place holonomic robots in different shapes based on their local state feedback and sensing [9]. Since their control law is given by the gradient of the implicit function which is represented by the zero isocontour of a 3D surface, the major limitation of this approach is concerned with local minima. Ikemoto, Hasegawa, Fukuda, and Matsuda described a gradual pattern formation approach for homogeneous robots to form various geometric shapes, in which a Turing morphogenetic function is proposed to generate patterns based on local information [10]. Under this approach, the final pattern is always generated from a circle pattern formed by robots in previous step. Thus, it is difficult to use this approach to form any repeated pattern.

In contrast to the above-mentioned methods, our algorithm accepts a graph representation of any pattern—including repeated patterns—as input.

Another body of related work solves multi-robot formation problems using task assignment approaches. Smith and Bullo demonstrated a provably-correct geometric target assignment method to allocate robots to distinct target locations as quickly as possible [11]. Michael, Zavlanos, Kumar, and Pappas proposed a distributed control approach for non-holonomic robots performing different formation tasks. They introduced a market-based coordination algorithm in which each robot determines its destination by bidding for task assignments [12]. Both algorithms require only local communication. Alonso-Mora, Breitenmoser, Rufli, Siegwart, and Beardsley proposed a control strategy to form arbitrary patterns, in which the goal positions for the final patterns are first generated via a Voronoi partition, after which a task assignment algorithm is performed to coordinate the robots [13]. However, the authors conducted centralized task assignment to compute an optimal formation, and used distributed controllers only for local collision avoidance. This approach does not seem applicable if only local information is available for each robot. Liu and Shell discussed a special gradual formation problem called "formation morphing," i.e., as the formation is gradually deformed in places, the formation changes but the major pattern remains unchanged [14]. They addressed this morphing problem in terms of the task assignment problem incorporated with the graph matching problem [15].

The primary limitation common to all of these task-assignment-based algorithms is that each robot needs to know its global coordinates and all the target positions in the global sense. Our algorithm offers each robot a way to compute task assignments and destinations using local sensing and communication in its local coordinate frame.

## III. Problem Statement

This section formalizes our problem. The details originally appeared in our prior work, and are restated here for completeness.

### A. Robots

We consider a collection of $n$ identical **robots** $R = \{r_1, \ldots, r_n\}$ moving through an obstacle-free plane.

#### 1) Robot identifiers

We assign each robot $r \in R$ a unique **identifier (ID)**, denoted $\mathrm{id}(r)$. We require only that, given the IDs $\mathrm{id}(r_i)$ and $\mathrm{id}(r_j)$ of two robots, any robot can **compare** $\mathrm{id}(r_i)$ to $\mathrm{id}(r_j)$. That is, robots should be able to consistently determine whether $\mathrm{id}(r_i) < \mathrm{id}(r_j)$ or $\mathrm{id}(r_j) < \mathrm{id}(r_i)$. In practice, robots' IDs could be assigned, for example, based on their serial numbers or network MAC addresses.

#### 2) Poses and coordinate frames

We represent the **pose** of each robot $r_i$ using $p(r_i) = (x_i, y_i, \theta_i)$ which consists of the robot's position and orientation, expressed in an arbitrary but fixed global coordinate frame.

In addition to this global frame, we also define a **body frame** attached to each robot, in which the robot is always at the origin, facing along the positive $x$-axis. Note that we use the global frame for modelling and explanatory purposes only; the robots do not know their own global coordinates in the environment. Thus, we denote the local pose of robot $r_j$ in the body frame of robot $r_i$ using $p_j^{(i)} = (x_j^{(i)}, y_j^{(i)}, \theta_j^{(i)})$.

We use a homogeneous matrix $T(r_i)$ to represent the rigid body transformation from the body frame of $r_i$ to the global frame:

$$T(r_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & x_i \\ \sin\theta_i & \cos\theta_i & y_i \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

*3) Motion, sensing, observation, and communication*

We assume that each robot can move forward with maximum linear velocity $v_{\max}$ and rotate in place with unbounded angular velocity.[1]

Let $\phi$ denote the robots' **range**, such that each robot can sense and communicate with other robots that are within distance $\phi$ of their own position. The other robots within this range are called **neighbors**. This sensing and communication occurs in discrete time steps, so that every $\Delta t$ seconds:

1) Each robot generates a collection of **observations**, indicating the IDs and relative poses of its neighbors.
2) Each robot broadcasts a **message** to its neighbors. Details about the specific messages used in our algorithm appear in Section V.

In practice, one can choose for $\phi$ either the robots' effective communication range or the robots' effective sensing range, whichever is smaller.

## B. Lattice graph

We use a directed graph to represent the desired pattern that the robots should form.

*Definition 1: 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 $w \xrightarrow{T(e)^{-1}} v$.*

The intuition is that, when the lattice is constructed, each robot will be associated with one lattice graph vertex, and that the outgoing edges of that vertex will correspond to the number and relative poses of that robot's neighbors.

*Definition 2: Given a lattice graph $G = (V, E)$ and a set of robots $R = \{r_1, \ldots, r_n\}$, we say that $R$* ***satisfies*** *$G$ if there exists a function $f : R \to V$ 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 $e_v^w : f(r_i) \longrightarrow f(r_j)$ in $E$, such that $f(r_i) = v$, $f(r_j) = w$, and $T(r_j) = T(r_i)T(e_v^w)$.*

That is, we require the transformation $T(e_v^w)$ associated with edge $e_v^w$ describe the relative pose of $r_j$ in the body frame of $r_i$. See Figure 2.

Notice that the definition of a lattice graph is general enough to include patterns that contradict themselves by

[1]The assumption of unbounded angular velocity is used to simplify the connectivity analysis in Section VI, but is not crucial to the operation of the algorithm.

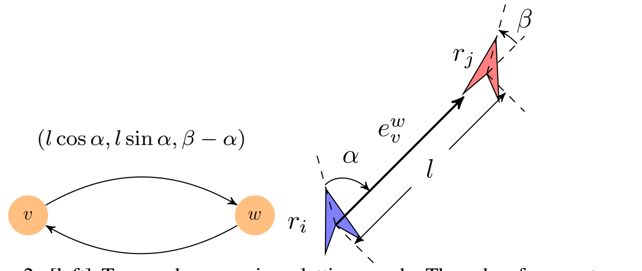
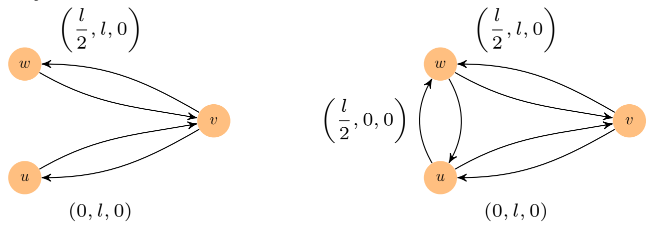


Fig. 2: [left] Two nodes $v, w$ in a lattice graph. The edge from $v$ to $w$ reflects the rigid body transformation between the two: a robot in role $w$ is at pose $(l\cos\alpha, l\sin\alpha, \beta-\alpha)$ relative a robot in role $v$. [right] Poses of robots $r_i$ and $r_j$ that satisfy this lattice graph, with role functions $f(r_i) = v, f(r_j) = w$.


$\left(\frac{l}{2}, l, 0\right)$
$w$
$v$
$u$
$(0, l, 0)$
$\left(\frac{l}{2}, l, 0\right)$
$\left(\frac{l}{2}, 0, 0\right)$
$w$
$v$
$u$
$(0, l, 0)$
Fig. 3: [left] A lattice graph that is not self-consistent when $\phi > l$. The distance between robots with roles $w$ and $u$ is less than $l/2 < \phi$, but no edge connects $w$ and $u$. [right] A lattice graph that is self-consistent when $\phi > l$.

forcing pairs of robots to be mutual neighbors, with roles that are not adjacent in the lattice graph. Figure 3 illustrates this idea, showing a comparison between a simple lattice graph that is not self-consistent with one that is self-consistent.

*Definition 3: 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 is less than or equal to $\phi$, there exist edges between $w$ and $u$.*

This definition is an important condition of the correctness proof in Section VI, because our algorithm is only correct when its input lattice graph is self-consistent.

## C. Evaluation criteria

We continue use the time and quality measurements proposed in our original paper [1] to evaluate both the time taken by our robots to form the desired formation, and the quality of that final formation.

The **execution time** $\bar{t}$ of the system is defined as the smallest time when all the robots reach static poses:

$$\bar{t} = \inf\{t \in (0, \infty) \mid \text{every } r \in R \text{ stays static after time } t\}$$

For robot $r_i$ with $N_i$ neighbors and associated with a lattice graph vertex with $E_i$ outgoing edges, we compute the as the fraction of its potential neighbors that are "missing," and average these values across all of the robots. We call the resulting value the **formation non-fulfillment ratio**:

$$\Gamma = \frac{1}{n}\sum_{i=1}^{n} \frac{E_i - N_i}{E_i} \quad (2)$$

The algorithm's objective is to drive the robots to a set of positions that satisfy a given lattice graph, while keeping both $\bar{t}$ and $\Gamma$ as small as possible.

## IV. Prior algorithm overview

Our prior work [1] introduced an algorithm for this problem. Because the new provably correct algorithm presented in this paper borrows many ideas from that antecedent, this section summarizes the existing algorithm. The basic idea is that at each time step, each robot exchanges messages with each of its neighbors. Each message contains:

1) a small data structure identifying the authority—that is, the level of importance—of the sender robot,
2) an integer identifying a lattice graph vertex as the role currently selected by the sender robot, and
3) a matching of the neighbors of the sender robot to out-edges of its role vertex, or to a dummy "no-match" element.

Based on the content of these messages, each robot decides which neighbor, if any, to accept a task assignment from—called the parent robot—and begins moving toward the location specified by the edge associated with itself in the parent's matching.

The algorithm proceeds statelessly, in the sense that each robot's decisions depend only on the messages that it has received and the neighbors is has observed in the most recent time step. Some detail about each step of the process appears below.

### A. Authorities

The specific concept of authority, which dictates when one robot should accept a task assignment from another robot, is based on the idea that the robot with the highest ID should act as the root of the formation, and that the other robots should organize around that root, using the relative distance to propagate task assignments outward.

*Definition 4: An* ***authority*** *is an ordered list of robot IDs*

$$\Lambda = \langle \mathrm{id}_1, \ldots, \mathrm{id}_k \rangle.$$

*The first ID* $\mathrm{id}_1$ *is called the* ***root*** *ID. The final ID* $\mathrm{id}_k$ *is called the* ***sender*** *ID. The number* $k$ *of IDs in the list is called its* ***length****.*

*Definition 5: Given two authorities* $\Lambda^{(1)} = \langle \mathrm{id}_1^{(1)}, \ldots, \mathrm{id}_k^{(1)} \rangle$ *and* $\Lambda^{(2)} = \langle \mathrm{id}_1^{(2)}, \ldots, \mathrm{id}_l^{(2)} \rangle$ *we say that* $\Lambda^{(2)}$ *is* ***higher than*** $\Lambda^{(1)}$*, denoted* $\Lambda^{(1)} < \Lambda^{(2)}$*, if*

1) $\mathrm{id}_1^{(1)} < \mathrm{id}_1^{(2)}$*, or*
2) $k < l$ *if* $\mathrm{id}_1^{(1)} = \mathrm{id}_1^{(2)}$*, or*
3) $\mathrm{id}_k^{(1)} < \mathrm{id}_l^{(2)}$ *if* $\mathrm{id}_1^{(1)} = \mathrm{id}_1^{(2)}$ *and* $l = k$*.*

In the first time step, each robot transmits an authority consisting of only its own ID. In subsequent stages, subject to a few minor restrictions, each robot selects as its parent the neighbor whose message contained the highest authority among all of its neighbors. If no neighbor has a higher authority than the robot's own ID as a singleton authority, then the robot considers itself a root, and does not choose any parent. Each descendant robot then generates an authority to transmit with its next message by appending its own ID to the authority of its parent. In this way, the robots form a collection of authority trees, one for each connected component of the communication graph, rooted at the robot in each such component with the highest ID.

### B. Role selection and task assignment

Each root robot selects—without loss of generality—the first vertex of the lattice graph as its role. Each descendant robot is assigned a role by its parent in the following way: The parent robot forms a matching between its neighbors and the out-edges of its role vertex in that lattice graph.

*Definition 6: Given a robot* $r_i$ *and a role vertex* $v$ *for that robot, let the lattice graph edge set* $O = \{\emptyset, e_v^w, e_v^u, \ldots\}$ *be the set that contains a null value* $\emptyset$ *and all outgoing edges from vertex* $v$*. Let* $I = \{\mathrm{id}_a, \mathrm{id}_b, \ldots\}$ *be the set that contains the IDs of the neighbors of* $r_i$*. Then a* ***matching for*** $r_i$ *is a function* $g : I \to O$ *that associates each neighbor ID with either a lattice graph edge from its role vertex or with the null value.*

Since this sort of matching is included in the message sent from parent to child, every descendant robot can select its role by identifying the target vertex of the edge matched to it by its parent. To generate a matching for its outgoing message, each robot uses the Hungarian algorithm [16] to match actual neighbors to desired neighbor positions, based on a minimization of total Euclidean distance. Any robot $r$ that the Hungarian algorithm does not match to any out-edge is assigned the null value $g(r) = \emptyset$.

### C. Movement strategy

Root robots do not move. Each descendant robot moves toward the location assigned to it in its parent's matching. Specifically, to ensure that it stays within communication range of its parent, robot $r_i$ moves to the point $p_i(t + \Delta t)$ closest to its final destination satisfying this bounded movement condition [1].

*Lemma 7:* If robot $r_i$ is the neighbor of robot $r_p$ at time $t$, and both robots move with maximum velocity $v$, it will still be a neighbor of $r_p$ at time $t + \Delta t$ if

$$||p_i(t + \Delta t) - p_p(t)|| \leq \phi - v\Delta t.$$

Descendant robots who are not matched to any edge by their parents move away from their parents and discard any incoming messages until they are outside the range of their parents.

## V. Algorithm description

This section introduces our algorithm for this problem. It proceeds with the same basic approach as described in Section IV, but with major changes in the authority tree construction (Section V-A), task assignment (Section V-B), and orphan handling (Section V-C) parts of the algorithm.

### A. Authority tree construction

A major limitation of our prior algorithm is that the procedure to construct an authority tree was heavily influenced

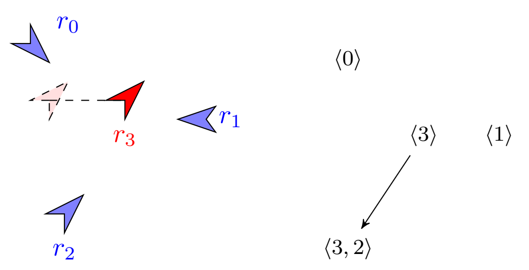


Fig. 4: [left] A root robot $r_3$ has three neighbors, but only one outgoing edge from its role. [right] The authority tree formed after exchanging messages. Robot $r_3$ selects $r_2$ as its child since $\text{id}_2$ is the closest ID among those IDs lower than $\text{id}_3$. Robots $r_0, r_1$ are not matched by $r_3$, regardless of their positions. Even as the robots move within the neighborhood of $r_3$, the authority tree remains the same.

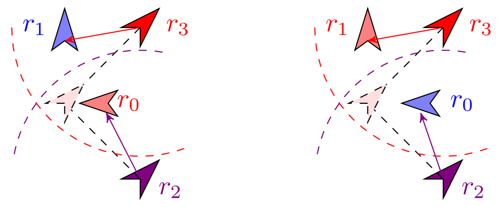


Fig. 5: [left] Robot $r_3$ forms a matching with $r_0$ since $r_0$ is closer to the target destination. [right] Robot $r_3$ forms a matching with $r_1$ since it has higher ID than $r_0$, when the distances between the target position to $r_1$ and $r_0$ are equal. At this moment, $r_3$ and $r_2$ assign $r_1$, $r_0$ to the same destination temporarily because $r_1$ is not observable to $r_2$. Once both $r_0$ and $r_1$ are observable to $r_2$ and $r_3$, then $r_3$ and $r_2$ both form matching with $r_1$, the assignment collision can be resolved.

by the robots' positions, since each robot forms its matching based to the relative locations of its neighbors. Thus, small position changes can lead to combinatorial changes in the authority tree, making it challenging to guarantee that the system will not become trapped in a cycle of authority tree reconfigurations.

In order to find a strategy to construct authority focusing on only robots' combinatorial relationship, we add an additional step in which each robot chooses which neighbors to match to its edges based on their IDs, rather than on their positions. That is, robots select as possible descendants the neighbors with the highest IDs less than its own. The number of neighbors selected as possible descendants does not exceed the number of outgoing edges of the robot's role in a given lattice graph. After using this procedure to select the neighbors to match, the robot continues as before, applying the Hungarian algorithm to assign tasks. The size of the cost matrix becomes $\min(N, E) \times E$, for a robot with $N$ neighbors and $E$ outgoing edges from its role. Each row corresponds to an ID of a non-orphan neighbor, and each column corresponds to an out-edge from its role.

Likewise, each descendant chooses for its parent the highest authority neighbor *that includes it in its matching*, rather than simply its neighbor with the highest authority. If there is not such neighbor to select as a parent, we call the robot an **orphan** if it is unmatched ($g(r) = \emptyset$) by every neighbor who has higher authority than its own.

See Figure 4. In combination, these changes facilitate the proof in Section VI-B that the authority tree eventually reaches a stable configuration.

### B. Task assignment

The authority tree has an undesirable side effect that, under the right conditions, two neighbors with different parents may be assigned to the same position.

To prevent this kind of failure, we add one additional refinement to the formation of the cost matrix for the Hungarian algorithm. If a robot observes that more than one of its neighbors are close to both one another and to the position associated with one of its outgoing edges, then this robot selects the neighbor closest to that position and forces its matching to that lattice edge and removes that robot and that edge from the task assignment instance. Figure 5 shows an example, in which the roles of both $r_2$ and $r_3$ have only one outgoing edge, both of which correspond to the same target position.

### C. Dealing with orphans

Another issue with the previous method is that the robots who have not been assigned a destination can wander aimlessly, and often become disconnected from other robots.

To ensure that each robot eventually reaches a useful position in a connected lattice, we add an new data member to each message indicating the "nearest opening" that the sending robot knows about. More precisely, an opening is a position where is not assigned to any robot. This occurs when some robots have fewer neighbors than out-edges, generally at the (external or internal) boundary of the formation. Openings can be found directly by observation (if the number of a robot's neighbors is less than the number of outgoing edges of its role) or indirectly via communications (by choosing the one where is closest to it after comparing all the positions included in its received messages).

To utilize this information, any orphan robot—one that is not matched by any potential parents—uses the nearest known opening position as its ultimate destination. To ensure that it makes progress toward this goal without becoming disconnected from the formation, it treats the robot that passed the nearest opening position to it as its "pseudoparent" and follows the bounded movement prescribed by Lemma 7 applied to this pseudoparent.

## VI. Algorithm Correctness

Recall that the objective of our approach is to enable all of the robots to reach positions satisfying the given lattice graph within bounded time. In this section, we prove that the algorithm described in Sections IV and V correctly achieves this goal.

The argument has three main parts. First, in Section VI-A, we define the combined communication-authority graph, and describe a total order over the space of authority graphs.

Next, in Section VI-B, we argue that, as the robots execute our algorithm, the changes in the authority graph monotonically increase with respect to this total order. Since there are only finitely many such graphs, this implies that the combined authority graph eventually becomes stable. We complete the proof in Section VI-C by demonstrating that, if the authority graph is stable, then the robots will reach positions that satisfy the lattice graph. These two arguments

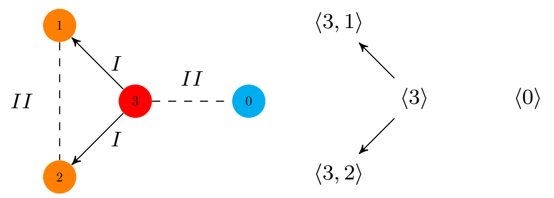


Fig. 6: [left] An example combined communication-authority graph. Solid directed edges, labeled $I$, indicate parent-child relationships. Dashed undirected edges, labeled $II$, belong to the communication graph but not the authority graph. [right] Two authority trees extracted from the combined graph on the left: $[(r_3, r_3), (r_2, r_3), (r_1, r_3)]$ and $[(r_0, r_0)]$. The authorities of robots corresponding to the left graph are: $\langle 0\rangle, \langle 3, 1\rangle, \langle 3, 2\rangle, \langle 3\rangle$, respectively.

rely on a common assumption on the connectedness of the communication graph. In Section VI-D, we show that our algorithm can guarantee the communication graph to remain connected during the algorithm execution.

## A. Total Order over Communication-Authority Graphs

We can use a graph structure to capture all the relevant information about the algorithm's progress.

*Definition 8: Given a set of robots $R = \{r_1, ..., r_n\}$, the* ***communication graph*** *is a graph in which each vertex is a robot, and there exists an undirected edge between two vertices if corresponding robots are neighbors. The* ***authority graph*** *is a graph in which each vertex is a robot, and there exists an directed edge to each descendant robot to its parent.*

For convenience, we draw the communication and authority graphs together, since they share the same set of vertices. The result, which we call a **combined communication-authority** graph, contains directed edges from the authority graph, which are labeled using $I$, and undirected edges labeled using $II$ from the communication graph. Figure 6 shows an example.

In the combined graph, for any two vertices, if there exists any path composed of edges from the communication graph between them, we use the term **steps** to denote the number of edges in the shortest path between them.

Next, we define a total order over combined graphs. Given two connected combined communication-authority graphs $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$ formed by the same set of robots $R$, we can define a total order, denoted $\leq$, as follows.

Define the **neighborhood properties** of a robot as: (1) the ID of its parent, (2) the number of its children, and (3) the lowest ID of its children. Identify the robot with the highest authority in $\mathcal{G}^{(2)}$ for which the neighborhood properties differ from its neighborhood properties in $\mathcal{G}^{(1)}$, and call it the **pivot robot**.

We say that $\mathcal{G}^{(1)} \leq \mathcal{G}^{(2)}$, if, for the pivot robot:

1) the ID of its parent in $\mathcal{G}^{(2)}$ is greater than the ID of its parent in $\mathcal{G}^{(1)}$, otherwise,
2) the number of its children in $\mathcal{G}^{(2)}$ is greater than the number of its children in $\mathcal{G}^{(1)}$, otherwise,
3) the lowest ID of its children in $\mathcal{G}^{(2)}$ is greater than the lowest ID of its children in $\mathcal{G}^{(1)}$.

The intuition of this ordering is that we want our algorithm to produce authority graph for the system monotonically so

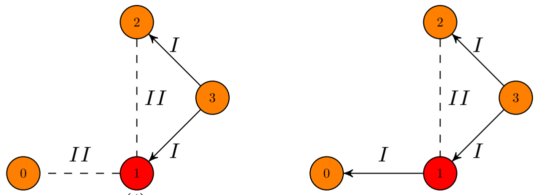


Fig. 7: [left] In $\mathcal{G}^{(i)}$, Robot $r_0$ and robot $r_1$ are neighbors. [right] In $\mathcal{G}^{(i+1)}$, Robot $r_1$ and robot $r_0$ form parent-child relationship. Robot $r_1$ is the pivot robot.

that finally we have a stable authority tree built. Therefore, we define the ordering above to evaluate the robots who are considered more important than others if they are closer to the root.

## B. Bounded-time Authority Tree Construction

In this section we analyze the changes that can occur in the combined communication-authority graph as the algorithm executes, arguing that every change leads to a new combined communication-authority graph that is no worse than the previous one. It happens that an action of a robot may cause multiple changes in the communication-authority graph at one moment. It is complicated to enumerate all the changes. Since we realized this can be seen as the degeneracy case, to simplify the analysis, we consider changes of a single edge at a time. Let $\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(i)}$ denote the sequence of such graphs, each differing from the previous by a single edge change. The possibilities for this single edge change can be enumerated in three cases:

1) It is possible for an edge labeled $II$ to become an edge labeled $I$, if (1) two robots were neighbors at time $t_i$, (2) one has higher authority than the other and it does not put the other one into his unmatched list, and (3) the number of neighbors is less than the out-degree of its role in the lattice graph. See Figure 7. In that case, in $\mathcal{G}^{(i+1)}$, the parent robot is the pivot robot more child that in $\mathcal{G}^{(i)}$, so that $\mathcal{G}^{(i)} \leq \mathcal{G}^{(i+1)}$.
2) It is possible for an edge labelled $I$ to become an edge labelled $II$, if (1) two robots were neighbors and their authorities shown a parent-child relationship at $t_i$, and (2) the number of non-orphan neighbors of this parent robot is equal to the out-degree of its role in the lattice graph. At time $t_{i+1}$, the parent robot meets a neighbor whose ID is higher than the ID of its child robot at $t_i$. Then the parent robot chooses this new neighbor as its child, the parent-child relationship is removed from the previous parent robot. See Figure 8. In this case, the parent robot is the pivot robot. Within the neighborhood properties of this pivot robot, the lowest ID of its children is higher in $\mathcal{G}^{(i+1)}$ than in $\mathcal{G}^{(i)}$. Therefore, $\mathcal{G}^{(i)} \leq \mathcal{G}^{(i+1)}$.
3) It is possible for an edge labelled $I$ to become an edge labelled $II$, if (1) two robots were neighbors and their authorities shown a parent-child relationship at $t_i$, and (2) the number of children of this parent robot is equal to the out-degree of its role in given lattice graph.
4) Finally, it is possible for edges labelled $II$ to appear or disappear from $\mathcal{G}^{(i)}$ to $\mathcal{G}^{(i+1)}$, without impacting the

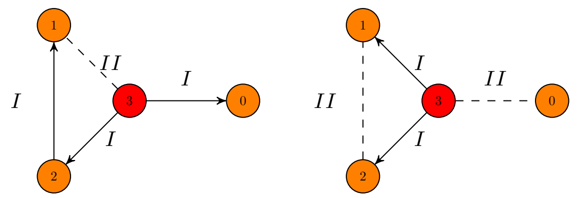


Fig. 8: An example in which the out-degree of each robot's role in the lattice graph is 2. [left] In $\mathcal{G}^{(i)}$, The child-parent relationships include $(r_1, r_2), (r_2, r_3), (r_0, r_3)$. [right] In $\mathcal{G}^{(i+1)}$, Robot $r_1$ and robot 2 are not a child-parent pair but only neighbors, meanwhile, robot $r_3$ forms a new parent-child relationship with robot 1. So the child-parent relationships evolve to $(r_1, r_3), (r_2, r_3)$. Robot $r_3$ is the pivot robot, and its lowest child ID improved in this change.

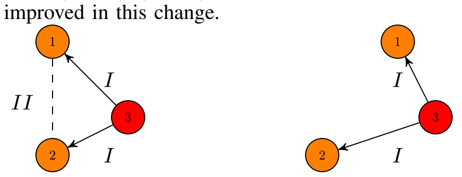


Fig. 9: [left] Robots $r_1$ and $r_2$ are neighbors and have matchings to each other, and they have common parent $r_3$. [right] Robots $r_1$ and $r_2$ are no longer neighbors due to their movements. The authority graph does not change either the combined graph changes from left to right (edge labeled with $II$ disappears) or from right to left (edge labeled with $II$ appears).

authority tree. For a given pair of robots, this can occur in two distinct ways: Either, (1) they were neighbors at $t_i$ but there was no parent-child relationship established between them, then at $t_{i+1}$, they are no longer neighbors due to their movements; or (2) they were not neighbors at $t_i$, at $t_{i+1}$ they become neighbors by sensing each other but no child-parent relationship, which can only be formed after they exchange messages in between. For either case, the communication graph changes but the neighborhood properties of each robot in the communication-authority graph do not change. Therefore, the authority graph at time $t_{i+1}$ is as the same as the authority graph at $t_i$. (Figure 9).

To sum up, as the robots execute our algorithm, whenever the communication graph changes, the authority graph will either remain the same or change to a new authority graph not worse than old ones. For a connected combined communication-authority graph composed of $n$ robots, there are $O(n!)$ possible authority graphs that can be constructed. That is, for an authority graph, there are $O(n!)$ possible changes it can undertake from the beginning. Moreover, an combined communication-authority graph at one time can never change back to be any of the previous ones. Finally the authority graph will not change when there is no change in the communication graph.

Taken as a whole, this analysis implies the result we sought.

*Lemma 9:* Given a set of robots $R = \{r_1, ..., r_n\}$, if the communication graph is connected, then the robots will form a stable authority tree, in which the number of matchings of each robot is not greater than the out-degree of the its role in given self-consistent lattice graph $G(V, E)$, within a finite number of steps.

## C. Correct positions given a stable authority tree

*Lemma 10:* Given a connected set of robots $R = \{r_1, \cdots, r_n\}$ and a self-consistent lattice graph, if those robots have formed a single stable authority tree, then the robots who have matching will move to positions satisfying given a self-consistent lattice graph $G$ in a finite number of steps.

*Proof:* Given an authority tree $\Lambda^\dagger$ that does not change as the algorithm executes, let $h$ denote the height of $\Lambda^\dagger$. Use induction on the height of $\Lambda^\dagger$.

First, lemma is true for the base case in which $h = 0$. In this case there is only one robot, a root which will always stay at its position. Thus, for $h = 0$, only 0 are needed steps to reach the destination position. Since no robot has any neighbors, this position vacuously satisfies the lattice graph.

For the inductive step, assume that the lemma is true for any authority tree that has height $h$: for any robot at the tree depth of $h$, it will move to the position satisfying the lattice graph in finite steps and then stop moving. Consider an authority tree $\Lambda^{\dagger^*}$ with height $h + 1$. Since we are only considering non-orphans any robot $r_c$ at depth of $h + 1$ of the authority tree, must have a neighbor at depth of $h$ of $\Lambda^{\dagger^*}$, be the parent of $r_c$, denoted as $r_p$, such that the edge $(r_c, r_p)$ is in $\Lambda^{\dagger^*}$. It suffices to show that any $r_c$ at depth $h + 1$, it move to a position satisfying the lattice graph in a finite number of steps.

Since the motion of $r_p$ is not impacted by any robot at depth $h + 1$, we know by inductive hypothesis that $r_p$ will reach a static position in a finite number $t$ of steps. At this time step, $r_c$ computes its destination $\bar{p}(r_c)$ in its local frame that satisfies the lattice graph edge following the assignment of $r_p$. Since $r_p$ is not moving, $r_c$ can reach that destination in a number of steps proportional to its linear velocity.

For any neighbor $r_n$ of $r_c$, who is neither its parent nor its child, note that $r_n$ has a parent at tree depth at most $h$, so it takes time $\Delta t_n$ to reach a static position after its parent is static. If $r_c$ and $r_n$ are not neighbors after they reach static positions, then the poses of $r_c$ and its parent satisfy the lattice graph, meanwhile the poses of $r_n$ and its parent satisfy the lattice graph. If $r_c$ and $r_n$ are still neighbors after they reach static positions, then they must have a common ancestor in $\Lambda^\dagger$, reachable along paths whose edges, by inductive hypothesis, match lattice graph edges. Therefore, recalling Definition 3, there are must exist edges between $f(r_c)$ and $f(r_n)$ so that the static positions of $r_c$ and $r_n$ satisfy the given self-consistent lattice graph. Therefore, for any robot $r_c$ and its neighbor $r_n$ at tree depth $h + 1$, it takes $\Delta t = \max(\Delta t_c, \Delta t_n)$ time for them to reach poses satisfying the self-consistent lattice graph after their parent(s) at tree depth $h$ reach the static position(s). $\square$

## D. Communication Graph Connectivity

The previous proofs rely on the idea that the robots' communication graph remains connected throughout the algorithm's execution. We now show that this is indeed that case.

*Lemma 11:* Given a set of robots $R = \{r_1, \ldots, r_n\}$, if the communication graph is connected at time $t$, then using our motion strategy, the communication graph stays connected at time $t + \Delta t$.

| | Hex-square | Oct-square | Tri-Hex-Square |
|---|---|---|---|
| **With 15 robots** | 0.380 | 0.412 | 0.390 |
| **With 30 robots** | 0.310 | 0.250 | 0.217 |
| **With 50 robots** | 0.223 | 0.220 | 0.215 |

Fig. 10: Average non-fulfillment ratios of forming three lattice patterns using the new algorithm.

| | Hex-square | Oct-square | Tri-Hex-Square |
|---|---|---|---|
| **With 15 robots** | 580 | 476 | 521 |
| **With 30 robots** | 1241 | 981 | 1103 |
| **With 50 robots** | 2518 | 1789 | 1826 |

Fig. 11: Average simulation cycles of forming three lattice patterns using the new algorithm.

*Proof:* Omitted due to space limitations. □

## VII. Implementation and Results

We implemented our algorithm using ROS and provide simulation results to evaluate the performance of our algorithm. In the experiments, we tested our algorithm with three lattice patterns: octagon-square, hexagon-square, and triangle-hexagon-square. The first one was introduced in our previous paper [1], and the latter two are newly created.

Here we consider scenarios involving 50 robots. For each lattice pattern, we run both the new and previous algorithm for 10 trials given initial pose distribution of robots generated using distinct seeds. The initial communication graph is provided connected.

Figures 11 and 10 show the statistics of the experiment results using the new algorithm. The average execution time is obtained by counting simulation cycles, the results show that the execution time scales as the number of robots increases. From our above proofs, we can say in fact that the larger the depth of the authority tree, the longer time required for all robots to reach static positions. The reason is that the algorithm is a local strategy, there exists a chain reaction during its execution: when a parent robot rotates, all of its descendant robots rotate. The frequent chain reactions bring side effect for the orphan robots to find the nearest opening, recall that our algorithm is stateless and an opening position is computed relative to robots headings, so the nearest opening position found by an orphan may change frequently due to the rotations of other robots. The average non-fulfillment ratios for different lattice pattern are very close, also for different number of robots. the reason is that our new algorithm guarantees the connectedness of the robots communication, so for each robot, the non-fulfillment ratio is lower-bounded.

Meanwhile, the formation quality formed by the new algorithm is improved compared with the old method. Since after executing the old algorithm, some robots are disconnected with others, which pulled up the non-fulfillment ratio of the whole system.

## VIII. Conclusion and Future Work

We presented a distributed multi-robot formation algorithm using only robots' local information based on our previous work. We proved that our algorithm guarantees to have robots reach positions satisfying the given lattice graph

| | Hex-square | Oct-square | Tri-Hex-Square |
|---|---|---|---|
| **New algorithm** | 0.223 | 0.220 | 0.215 |
| **Old algorithm** | 0.433 | 0.340 | 0.378 |

Fig. 12: Comparison of the average non-fulfillment ratios using two algorithms on three lattice patterns formation, with 50 robots.

within finite steps. Moreover, our algorithm maintains the connectedness of the communication graph during its execution. Compared to the prior algorithm, the new algorithm runs slower but brings better formation quality in general.

Realized that the execution time of our algorithm scales as the number of robots increases. We focus on improving the method to get better performance, especially, to avoid the chain reaction. Last but not least, it would be interesting to extend the algorithm from 2D plane to a high dimension space.

## References

[1] Y. Song and J. M. O'Kane, "Decentralized formation of arbitrary multi-robot lattices," in *Proc. IEEE International Conference on Robotics and Automation*, Hong Kong, China, 2014.

[2] K. Fujibayashi, S. Murata, K. Sugawara, and M. Yamamura, "Self-organizing formation algorithm for active elements," in *Reliable Distributed Systems, 2002. Proceedings. 21st IEEE Symposium on*. IEEE, 2002, pp. 416–421.

[3] W. M. Spears, D. F. Spears, J. C. Hamann, and R. Heil, "Distributed, physics-based control of swarms of vehicles," *Autonomous Robots*, vol. 17, no. 2-3, pp. 137–162, 2004.

[4] E. Martinson and D. Payton, *Lattice formation in mobile autonomous sensor arrays*. Springer Berlin Heidelberg, 2005, vol. 3342, ch. Lecture notes in computer science, pp. 98–111.

[5] I. Navarro, J. Pugh, A. Martinoli, and F. Matía, "A distributed scalable approach to formation control in multi-robot systems," in *Distributed Autonomous Robotic Systems 8*. Springer, 2009, pp. 203–214.

[6] S. Lee, A. Becker, S. P. Fekete, A. Kröller, and J. McLurkin, "Exploration via structured triangulation by a multi-robot system with bearing-only low-resolution sensors," *arXiv preprint arXiv:1402.0400*, 2014.

[7] S. Prabhu, W. Li, and J. McLurkin, "Hexagonal lattice formation in multi-robot systems," in *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, 2012.

[8] R. J. Mullen, D. Monekosso, S. Barman, and P. Remagnino, "Reactive coordination and adaptive lattice formation in mobile robotic surveillance swarms," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 229–242.

[9] L. Chaimowicz, N. Michael, and V. Kumar, "Controlling swarms of robots using interpolated implicit functions," in *Proc. IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 2487–2492.

[10] Y. Ikemoto, Y. Hasegawa, T. Fukuda, and K. Matsuda, "Gradual spatial pattern formation of homogeneous robot group," *Information Sciences*, vol. 171, no. 4, pp. 431 – 445, 2005, intelligent Embedded Agents.

[11] S. L. Smith and F. Bullo, "A geometric assignment problem for robotic networks," in *Modeling, Estimation and Control: Festschrift in Honor of Giorgio Picci on the Occasion of his Sixty-Fifth Birthday*, A. Chiuso, A. Ferrante, and S. Pinzoni, Eds. Springer, 2007, vol. 364, pp. 271–284.

[12] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Distributed multi-robot task assignment and formation control," in *Proc. IEEE International Conference on Robotics and Automation*, 2008.

[13] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Multi-robot system for artistic pattern formation," in *Proc. IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4512–4517.

[14] L. Liu and D. A. Shell, "Multi-robot formation morphing through a graph matching problem," in *International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Baltimore, MD, Nov 2012.

[15] L. Lovasz, *Matching Theory (North-Holland mathematics studies)*. Elsevier Science Publisher Ltd, 1986.

[16] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955. [Online]. Available: http://dx.doi.org/10.1002/nav.3800020109