# Hexagonal Lattice Formation in Multi-Robot Systems

Paper 128


## ABSTRACT

We present an algorithm that arranges a multi-robot system into a regular hexagonal lattice. This configuration provides continuous coverage with the fewest number of robots required. It also has a bounded stretch over a fully-connected graph, producing an efficient multi-hop communications network. Our algorithm uses artificial forces to move each robot to local potential energy wells. A local error correction algorithm detects and corrects most local lattice errors. Both algorithms are fully distributed, requiring *local network geometry* information, but no global coordinates. We present analysis of the potential energy wells that form the lattice, a proof of the upper bound on the spanning ratio of a hexagonal packing, and the error detecting and correcting algorithm. Simulation results demonstrate the effectiveness of the approach for large populations of robots.



## Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Distributed Applications

## General Terms

Algorithms

## Keywords

distributed algorithms, multi-agent networks, full coverage, surveillance, sensor networks, lattice formation, collective intelligence, distributed problem solving


## 1. INTRODUCTION

Key applications of multi-robot systems, like mapping, exploration, search-and-rescue, and surveillance, require robots to disperse over large geographical area while maintaining a communications network. Some of these applications require continuous coverage of areas, efficient network connectivity, and the efficient allocation of each robot. This work accomplishes this task with a distributed algorithm that positions the robots at the vertices of a hexagonal lattice. Figure 1 shows the results of our algorithm run in simulation with 600 robots.

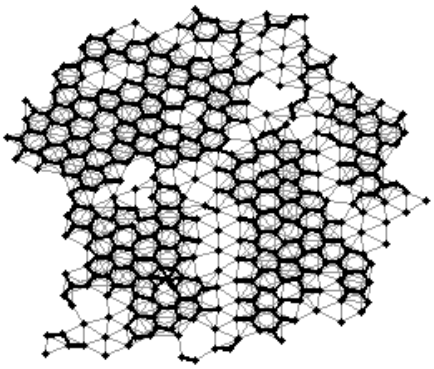
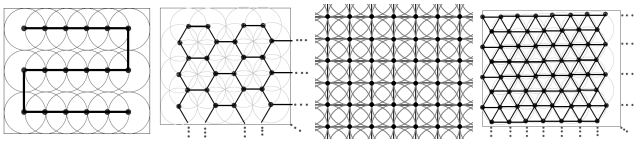


**Figure 1: An example hexagonal lattice from a simulation run of our algorithm.**

The hexagonal configuration requires the fewest robots per unit area of any regular tessellation pattern and produces a connected communication network with a bounded stretch. Figure 2 shows four alternatives for coverage, a line of robots, and the three regular tessellations: hexagonal, square, and triangular lattices. All three regular tessellations produce a communication graph with a bounded *stretch*, which is the ratio between the distance a message must travel through the network between two robots and the Euclidean distance between these same two robots. A constant stretch ensures that the distance a message has to travel depends only on the euclidean distance. Therefore, we can increase the number of robots without increasing the distance the message has to travel. The line of robots provides full coverage with the smallest number of robots, but produces a communications network with a stretch that is $O(n)$, where $n$ is the total number of robots in the network. Of these, the hexagonal lattice configuration provides total coverage of an area, uses the least number of robots, and produces a network with a bounded stretch.

There are four contributions in this paper. We present an algorithm that uses *virtual forces* between two species of robots to produce a hexagonal cell, and we show



(a) Line Graph (b) Hexagonal Lattice (c) Square Lattice (d) Triangular Lattice

**Figure 2: Four different communication network topologies for full coverage, sorted by the density per unit area. The line graph uses the smallest number of robots, but has graph stretch of $O(n)$. The triangular and square lattice have bounded stretch, but uses more robots than necessary. The hexagonal lattice also has bounded graph stretch, but requires the fewest number of robots for full coverage, only $\frac{2}{3}$ of the robots from a triangular lattice.**

that these same forces will allow the hexagonal cell to propagate into a hexagonal lattice. We show that the potential energy field around the robots has a local minimum that causes errors in the lattice, but the error can be detected and corrected with a second distributed algorithm. We present a proof sketch of the maximum stretch of the spanning graph induced by the lattice, and empirical data that the average stretch is actually much lower. Finally, we demonstrate the results of simulations with large populations of 400 robots. All of the algorithms are fully distributed, scale to large numbers of robots, and require only the *local network geometry - i.e.* the positions of neighboring robots in the communications network. The algorithms require limited inter-robot communication, and are designed to run on simple, low-cost mobile robot platforms.

## 1.1 Related Work

Our hexagonal lattice algorithm builds on the work of Spears et al. Using artificial forces, they developed formation control algorithms to construct triangular[1] and square lattices [5,10–12]. We modified their framework to make hexagonal cells stable, and we developed an algorithm to remove lattice imperfections.

Triangular lattices provide complete coverage but incur a great deal of overlap in the communication ranges of neighboring robots. As shown in Figure 2, a hexagonal lattice can be formed from a triangular lattice by removing every third robot from each line. A virtual robot could replace a robot in this position as suggested by Mullen et al. [8], but we chose not to use this method because it would require coordination among several robots to determine the center of the hexagonal cells.

[1]There is an unfortunate habit in the literature of referring to triangular lattices as hexagonal lattices. In this work, we name a lattice by the polygon that forms it.

Our approach uses two species of robots, similar to the "spin up/down" idea of Spears et. al. to produce a potential energy field that has energy wells located at the vertices of a hexagonal cell.

Unfortunately, the physicomimetric framework often leads to robots settling down in unfavorable local minima. Martinson and Payton were able to avoid local minima in square lattices by using the robot's onboard compass to draw several parallel lines and making each robot attracted to the parallel lines [7]. Once on the parallel line, they only move on the line. This scheme avoids several local minima that may occur off the lines parallel to the two orthogonal axis in a square lattice. We could not employ this method in avoiding local minima while creating the hexagonal lattice because the hexagonal lattice is not formed by orthogonal lines, and, instead we developed another algorithm to remove a robot from a local minimum.

Previous work by Tucker Balch and Maria Hybinette shows the possibility of using potential fields to maintain a geometric formation while also avoiding obstacles. The robots constructed "social potentials," where each robot influenced the potential energy in its vicinity [2]. Desai et al. created an algorithm to maintain relative positions and construct a formation [3]. Our work differs from that of Balch and Hybinette and Desai et al. because they created a set of geometric formations; however, we create a repeating hexagonal lattice. Balch and Hybinette showed that by placing repulsive forces near obstacles, the robots could successfully avoid them. In future work, we can have the robots in the repeated pattern maneuver around obstacles.

While we focus on constructing a repeated hexagonal lattice, our algorithm shares some similarities to previous work in flocking. Blake Eikenberry et al. created an algorithm that allows a swarm to construct a formation. The robots detected nearby robots, constructed trajectories, and positioned themselves in the proper relative positions [1]. Our work differs from work done by Eikenberry et al. because they had to prescribe the relative positions for each robot. In our work, the robots determine their relative positions using swarm intelligence [6]. Hanada et al. constructed algorithms for separating triangular lattices at an obstacle and then reunifying them. However, our work is novel because they did not create a self-organized hexagonal lattice. Turgut et al. created algorithms for self-organizing the heading and proximity of the robots. However, our algorithm not only maintains a certain proximity, but it also constructs a repeated hexagonal lattice formation.

# 2. PROBLEM STATEMENT

We assume that we have $n$ randomly distributed robots that can measure their *local network geometry*, the posi-

tions of their neighbors in each robot's reference frame. A robot is able to move freely in any direction in the plane. We assume that each robot has a sensing radius of $R$ and a local communication radius of $1.5R$. We wish to implement this on low-cost systems with limited communications and processing, so care is taken to implement solutions that are simple and scalable. We seek to position the robots onto the vertices of a hexagonal lattice with hexagons that have a side length of $R$.

For communication, we assign $\log_2(n)$ bits to provide unique IDs to each robot, 1 bit to identify the type of robot, and 32 bits to communicate the error the robot senses. The total number of bits required is $\log_2(n)+33$

## 3. GRAPH STRETCH IN HEX LATTICE

Let $K$ be a complete graph where the nodes are arranged as the vertices of a hexagonal lattice. Let $H$ be the edges of the hexagonal lattice, and $G$ be the vertices of the hexagonal lattice. By definition, $H$ is a spanning tree of $G$. Define $d_h(A, B)$ as the shortest distance $A$ and $B$ on $H$ and $d(A, B)$ as the shortest distance between $A$ and $B$ on $K$.

Define the stretch as $\frac{d_h(A,B)}{d(A,B)}$. First, we use computational methods to produce a distribution of the stretch. Then, we prove the stretch is no greater than 1.5.

### 3.1 The Distribution of Stretch

To compute the distribution of stretch, we constructed a hexagonal lattice that is ten hexagons by ten hexagons and calculated the distribution of the stretch shown in Figure 3(c). The maximum ratio is 1.5, and this maximum corresponds to traveling between opposite sides of a hexagonal cell. Furthermore, the histogram indicates that the distribution of stretch resides mostly in the range 1.2-1.4.

### 3.2 Analytical Solution for Stretch

We wish to show

$$\frac{d_h(A,B)}{d(A,B)} \leq 1.5.$$

Let $(a, b)$ be the index of an arbitrary lattice point. WLOG choose one point to be the origin $(0, 0)$, restrict $a, b \geq 0$, and let side lengths equal one. Assign $(a, b)$ as shown in Figure 3(a).

Let $d(a, b)$ denote the Euclidean norm of $(a, b)$. Then

$$d(a, b) = \sqrt{x(a, b)^2 + y(a, b)^2}.$$

The $y$-position of point $(a, b)$ is

$$y(a, b) = y(b) = \tfrac{\sqrt{3}}{2}b.$$

The $x$-position depends on the parity of $a + b$:

$$x(a, b) = \left\{ \begin{array}{cc} \frac{3}{2}a - \frac{1}{2}, & \text{if } a + b \text{ odd} \\ \frac{3}{2}a, & \text{if } a + b \text{ even} \end{array} \right\}.$$

Let $d_h(a, b)$ denote the shortest walk from $(0, 0)$ to $(a, b)$ on the hexagonal lattice. Without altering walk lengths, we reshape the hexagonal lattice into a "brick" lattice as shown in Figure 3(b);

The complete derivation of $d_h$ is outside the scope of this discussion, so we present an outline:

If $b > a$, $d_h(a, b) \geq a+b$. A walk of length $a+b$ exists, so $d_h(a, b) = a + b$.

If $b \leq a$, we have two sub-cases. If $a + b$ is odd, then $d_h(a, b) \geq 2a - 1$. A walk of length $2a - 1$ exists, so $d_h(a, b) = 2a-1$. We similarly show for even $a+b$ that $d_h = 2a$.

In summary,

$$d_h(a, b) = \left\{ \begin{array}{cc} a + b, & \text{if } b > a \\ 2a - 1, & \text{if } b \leq a, a + b \text{ odd} \\ 2a, & \text{if } b \leq a, a + b \text{ even} \end{array} \right\}.$$

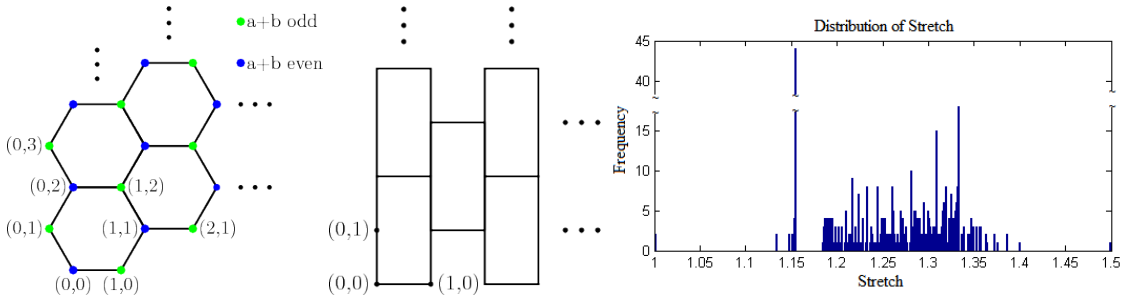
To finish the proof, we show $\frac{d_h}{d} \leq 1.5$ for each case. The algebraic details are outside the scope of this discussion.

## 4. LATTICE FORMATION

To create a stable hexagonal cell, we divide the population into two types of robots, red and blue. Discriminating between red and blue requires 1 bit, and the robots can be divided into red and blue types by assigning a 50% probability to being one of the colors. Robots of the same color try to maintain a distance of $\sqrt{3}R$ between each other while robots of different colors try to maintain a distance of $R$ between each other. Maintaining these prescribed distances will form the stable hexagonal cell pictured in Figure 4. The nucleus of a hexagonal cell is a triangle with one side of length $\sqrt{3}R$, and two sides of length $R$. This requires two robots of one color and one of the other. Assuming that we begin with 50% red and 50% blue robots and that three robots are interacting, the probability that two are of the same type and one is of another type is $\frac{3}{4}$ : $1 - P(\text{All Red}) - P(\text{All Blue}) = 1 - 2\left(\frac{1}{2}\right)^3 = \frac{3}{4}$. Across the entire population, this will generate a hexagonal nucleus with high probability. Our simulation results validate this assumption, the robots readily formed an initial nucleus.

The physicomimetric framework models the force between the robots as a proportionality constant, $G$, times the product of the masses divided by the distance raised to a power, $p$. A viscosity term, $c$, allows the robots to reach a stable state by having their velocity decay to zero. The framework also imposes a maximum force, $F_{max}$, that the robots can impart on each other to avoid a rapid increase in velocity.

We determined values for constants $F_{max}$, $G$, $p$, and $c$ by searching the parameter space with a genetic algorithm [4, 9]. We used a genetic algorithm to quickly tune parameters to optimize global properties of the




(a) Indexing Hexagonal Lattice Nodes (b) Brick Lattice (c) Distribution of Stretch

**Figure 3: a: The hexagonal lattice can be compressed into a "brick" lattice without changing walk lengths. The stretch is at most 1.5. b: The "brick" lattice conserves the indexes and edges from the hexagonal lattice c: We computed the ratio between the shortest path traveled and the Euclidean distance based on a large lattice. This lattice has a size of ten hexagons positioned vertically and horizontally for a total of 420 vertices. The largest ratio of 1.5 between the distance traveled along the lattice and the euclidean distance occurs in traveling between opposite sides of the same lattice as shown in the. There is a spike in the histogram at a stretch of 1.15, corresponding to the most common stretch between any two nodes.**

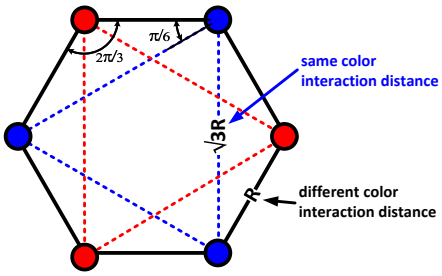



**Figure 4: In the hexagonal cell pictured above, the robots maintain the prescribed distances between each other, making the cell stable.**

lattice. We evolved parameters to minimize global lattice error, which we define later in this paper. Spears et al. tuned their parameters to reduce the number of clusters and to reach a phase transition state, where the robots in a cluster repel each other with more force than the force from the surrounding robots pushing them into a cluster.

## 4.1 Mass Interactions

To maintain a Euclidean distance R between robots of different types and a distance $\sqrt{3}$R between robots of the same type, we define two stable distances,

$$R_{stable} = \left\{ \begin{array}{ll} R, & \text{for robots of different types} \\ \sqrt{3}R, & \text{for robots of the same type} \end{array} \right\}.$$

By making the force attractive when the distance between the robots, $d$, is greater than $R_{stable}$, repulsive when $d$ is less than $R_{stable}$, and zero when $d$ equals $R_{stable}$, the robots will move to reach the desired distance between each other. To ensure that the interactions remain local, the force goes to zero for $d$ greater than $1.5R_{stable}$.

The magnitude of the force between two robots of mass $m$ is determined from the equation for the virtual force as defined in Spears et al. [10]:

$$F = \left\{ \begin{array}{ll} \frac{G}{d^p}, & \text{for robots of different types} \\ \frac{G}{(\frac{d}{\sqrt{3}})^p}, & \text{for robots of the same type} \end{array} \right\}.$$

For our simulation, we used $G = 627.2977$ and $p = 1.6185$.

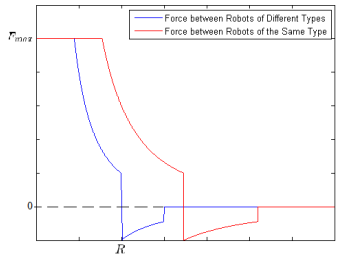
## 4.2 Stability

The virtual force approaches infinity as two robots reach a distance where d is very close to zero. When numerical methods are used to integrate an acceleration, resulting from a force that goes to infinity, the velocity can reach a very large value, causing instabilities in the system. Thus, we placed a maximum value on the magnitude of the virtual force, $F_{max} = 50.8168$. The total composite virtual force is shown in Figure 5(a).

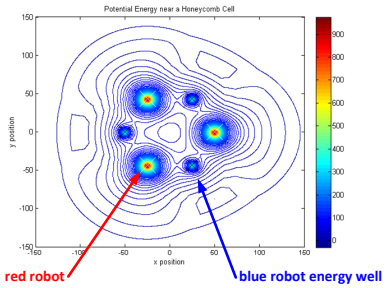
Additionally, we desire for the robots to lose some energy and eventually settle down into their stable positions. This goal is achieved by introducing a viscosity coefficient, $c = 0.2594$, that decreases the magnitude of the velocity.

Changes in the communication radius will not affect the algorithm's stability. The robot's communication radius can fluctuate between $R_{stable}$ and $1.5R_{stable}$, and the robot will still be able to detect the energy well at $R_{stable}$ and settle onto a vertex of the honeycomb lattice.

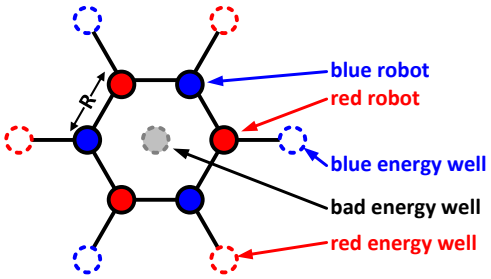
## 4.3 Lattice Propagation




(a) Force Function




(b) Potential Energy Contours




(c) Energy Wells Diagrams

**Figure 5: a) shows the force that two robots impart on each other. b) shows the potential energy wells for incoming red robots. Using symmetry arguments, we can add three more energy wells for blue robots, which are drawn in c).**

We also require that a hexagonal cell, once formed, will give rise to more hexagonal cells and eventually a hexagonal lattice. The potential energy diagram in Figure 5(b) indicates that there are four stable positions for an incoming red robot in the vicinity of an already assembled hexagonal cell. By symmetry, we can add three energy wells for blue robots, pictured in Figure 5(c). Figure 5(c) also shows that we can assemble significant portions of six adjacent hexagonal cells surrounding the one that has already formed. Thus, a single hexagonal cell acts as a nucleus for the creation of a hexagonal lattice. However, our potential energy field creates a problematic stable position in the center of the hexagonal cell, which causes an *interstitial error*, and causes the hexagonal lattice to become a triangular lattice. This energy well exists for all choice of constants because all forces in this location sum to zero. In order to create a hexagonal lattice, we must design a distributed algorithm to remove a robot caught in this undesirable location.

## 5. ERROR DETECTION AND CORRECTION

To remove the robot from the interstitial position at the center of a hexagonal cell, we developed a distributed algorithm that allows each robot to measure its local error and determine if it at the center of a hexagonal cell. If the robot is at the center of the hexagonal cell, it is moved to the exterior of the convex hull surrounding all the robots. The measurement and communication of error requires 32 bits.

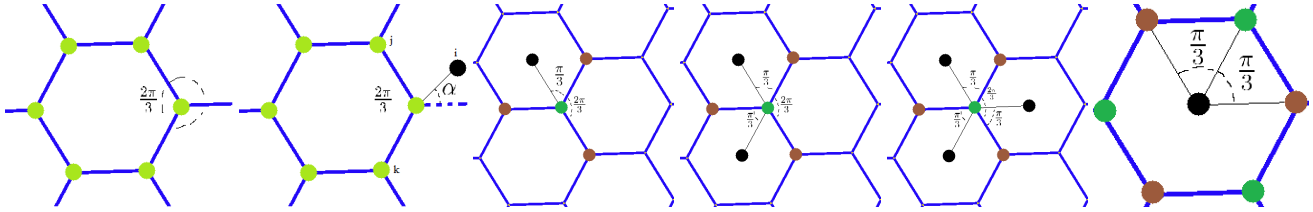
## 6. DEFINING ERROR

Figure 6(a) show a perfect hexagonal lattice. In this ideal configuration, the angle between any two neighbors, $\theta_{ij}$, is a multiple of $\frac{2\pi}{3}$. We define *local lattice error*, $E_{ij}$, for a neighbor pair $i, j$ as the smallest difference between $\theta_{ij}$ and a multiple of $\frac{2\pi}{3}$: $E_{ij} = min(mod(\theta_{ij}, \frac{2\pi}{3}), \frac{2\pi}{3} - mod(\theta_{ij}, \frac{2\pi}{3}))$. To calculate the total error of robot, we sum the error over all neighbor pairs $i, j$: $E_{tot} = \sum E_{ij}$. The average error of a robot is defined as the total error divided by the number of neighbor pairs: $E_{avg} = \frac{E_{tot}}{\Delta C_2}$, where $\Delta$ is the number of neighbors.

### 6.1 Error Configurations

We simplify this discussion by assuming that most of the robots are at the center of their energy wells, forming a hexagonal lattice. We break the discussion into several cases to illustrate how the error metric detects interstitial lattice defects.

**Case 0:** Figure 6(a) shows that a robot with all its neighbors positioned on the vertices of a hexagonal lattice will measure $\theta_{ij}$ between neighbors $i, j$ that are



(a) Perfect Hexagonal Lattice (Case 0) (b) Example Error Reading (Case 0) (c) Case 1 (d) Case 2 (e) Case 3 (f) Case 4

**Figure 6: Assuming that the only stable positions are on the vertices of or in the center of a hexagonal cell, the cases above show the possible neighborhoods a robot could find itself in.**

multiples of $\frac{2\pi}{3}$. This produces a total error $E_{tot} = \sum E_{ij} = 0$.

**Case 1:** Figure 6(c) shows a robot with three neighbors on the hexagonal lattice and one neighbor in an interstitial position. This will produce an error of $\frac{\pi}{3}$ between the brown neighbor and each of the other neighbors, and an error of zero between all other robots. Since there are three $\theta_{ij}$ that include the brown robot, the total error will be $E_{tot} = 3\left(\frac{\pi}{3}\right) = \pi$. Since there are a total of ${}_4C_2 = 6$ angles between all neighbor pairs, the average error for this robot will be $E_{avg} = \frac{\pi}{6}$

**Case 2:**Figure 6(d) shows a green robot with three neighbors on a hexagonal lattice and two neighbors in an interstitial position. The green robot will read an error of $\frac{\pi}{3}$ between brown and black neighbors, an error of zero between two black neighbors, and an error of zero between two brown neighbors. Since the robot will measure 6 $\theta_{ij}$ between brown and black neighbors, 1 $\theta_{ij}$ between two black neighbors, and 3 $\theta_{ij}$ between two brown neighbors, the robot will read a total error of $E_{tot} = 6\frac{\pi}{3} + 3(0) + 1(0) = 2\pi$. Since there are a total of 10 angle readings, this total error corresponds to an average error of $E_{avg} = \frac{\pi}{5}$.

**Case 3:**The green robot in case 3 is surrounded by three brown neighbors that are on a hexagonal lattice and three black neighbors in interstitial positions as pictured in Figure 6(e). The green robot measures an error of $\frac{\pi}{3}$ between brown and black neighbors, measures an error of 0 between two brown neighbors, and measures an error of 0 between two black neighbors. There are a total of 9 $\theta_{ij}$ with brown and black neighbors, 3 $\theta_{ij}$ between two brown neighbors, and 3 $\theta_{ij}$ between two black neighbors. Thus, the total error measured by the robot is $E_{tot} = 9\frac{\pi}{3} + 3(0) + 3(0) = 3\pi$. Since there are a total of 15 ij neighbor pairs, this total error reading corresponds to an average error of $E_{avg} = \frac{\pi}{5}$

**Case 4:**The black robot in case 4 positions itself in the interstitial position, and it will measure an error of $\frac{\pi}{3}$ between brown and green neighbors, zero error between two brown neighbors, and zero error between two green neighbors. Since there are a total of 9 $\theta_{ij}$ between brown and green neighbors, 3 $\theta_{ij}$ between two green neighbors, and 3 $\theta_{ij}$ between two brown neighbors, the robot will read a total error of $E_{tot} = 9\frac{\pi}{3} + 3(0) + 3(0) = 3\pi$. Since there are a total of 15 angles, this total error reading corresponds to an average error of $E_{avg} = \frac{\pi}{5}$

## 6.2 The Error Correction Algorithm

Figure 7 shows a configuration in which there is are many interstitial errors. This is effectively creating a triangular lattice within the hexagonal lattice. With this many errors, there is some boundary between the two lattice types. In such a configuration, our error correction algorithm can start with any interstitial robot that is inside a hexagonal cell that is adjacent to three or more correct hexagonal cells. Removal of this robot would extend the hexagonal lattice. We call this robot a *correction start point*, and have circled such a robot in black in Figure 7.

Figure 7 shows that all robots at the boundary of triangular and hexagonal cells have either case 1 or case 2 neighbors; furthermore, a correction start point has two case 1 neighbors that make an angle of $\frac{\pi}{3}$ with each other; we can discriminate a case 1 robot from a case 2 robot by using the criterion $E_{avg} \leq \frac{\pi}{6}$. Thus, if a robot reads an angle of $\frac{\pi}{3}$ between two neighbors with an average error less than $\frac{\pi}{6}$, it determines itself to be at a correction start point. Having determined itself to be correction start point, it moves to the exterior of the lattice. This process continues indefinitely, removing many of the interstitial errors in the network.

# 7. SIMULATION RESULTS

Figure 8 shows the position of the robots after running the entire algorithm for 2000 time steps. The hexagonal cells are fairly easy to pick out in both cases. To evaluate the global performance, we define lattice error as: $E_{lat} = \sum_{i=1}^{n} \frac{E_{avg,i}}{n}$, where $n$ is the number of robots and $E_{avg,i}$ is the $E_{avg}$ corresponding to the $i^{th}$ robot. Figure 9(b) shows that for a simulation run with

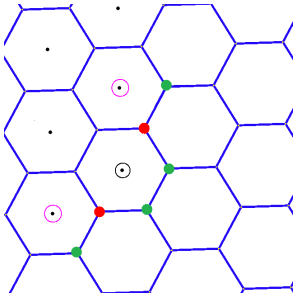


**Figure 7: The robot circled in black is at a convex point of the hexagonal lattice and measures an angle of $\frac{\pi}{3}$ between two case 1 neighbors, which are colored green. Upon removal of the robot circled in black, the robots circled in pink are at convex points.**

the error correction algorithm, the lattice error decreases up until the 500th time step; whereas, the lattice error for a simulation run without the error correction algorithm ceases to decrease very early in the simulation. Figure 9(c) shows that the error with error correction algorithm is always less than error without the algorithm. Figure 9(a) shows that the error correction algorithm has almost no effect on the average speed, indicating that the error correction algorithm does not cause instabilities in the system.

Figure 9(a) shows that the loss of messages does not cause the average speed to go up, indicating the system remains stable. Figure 9(b) shows that increasing the loss of messages decreases the error of the system run without the error correction algorithm. The decrease in error can be attributed to the loss of stability at the center of the hexagonal cells when one of the robots on the vertices fails to impart a force. However, increasing the loss of messages increases the error of the system run with the error correction algorithm. The increased error can be attributed to an inability to communicate error and fix the lattice.

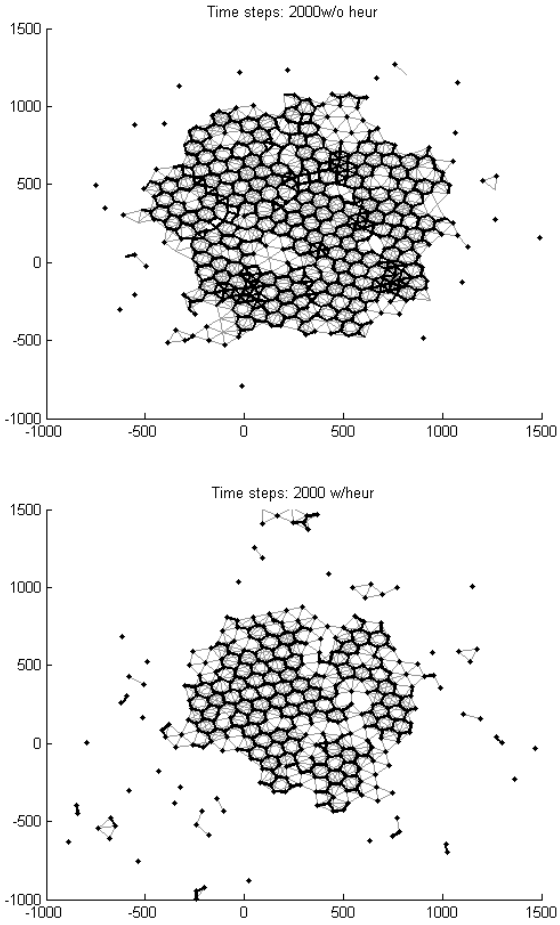
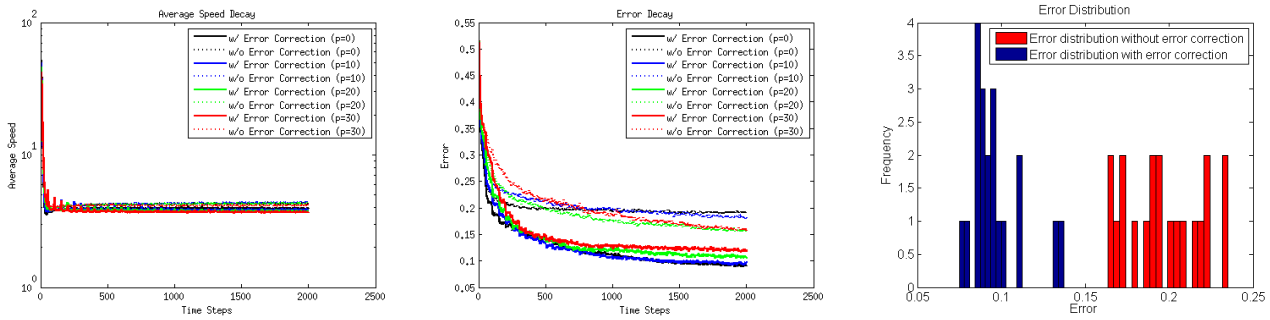



**Figure 8: The top diagram shows the position of the robots when the simulation is run without error correction, and the bottom diagram shows the result when the simulation is run with error correction.**

## 8. CONCLUSIONS AND FUTURE WORK

Because our algorithm is fully distributed, multiple nuclei start hexagonal lattices simultaneously, ultimately creating crystalline grain boundaries in the final structure. This is unavoidable in our current design, but future implementations could take advantage of symmetry-breaking techniques to break ties at grain boundaries, and have one lattice "rearrange" neighboring grains to produce a single-crystal lattice. Because our current implementation requires communication with neighbors at a range of $\sqrt{3}R$, we cannot form hexagonal cells at the maximum extent of the communication range. We think that it is possible to build a hexagonal lattice using a communication range of $R$ using virtual forces, but it will require a more complicated distributed al-




(a) Average Speed Decay

(b) Error Decay

(c) Error Distribution after 2000 Time Steps(p=0)

**Figure 9: In the plot above, we observe decaying speed and error when messages are dropped p percentage of the time. These diagrams show that as time evolves, the simulation with the error correction algorithm decays to a smaller error while also maintaining stability as evidenced by the decreasing average speed. They, also, show that a higher probability of message loss decreases lattice error when the error correction algorithm is not applied. However, with the error correction algorithm, a higher probability of message loss increases lattice error.**

gorithm to form hexagonal cells, an approach we will explore in future work. Our current algorithm is simple, requires only local network geometry, and is effective. We have calculated fundamental graph properties of the hexagonal lattice. The simulation results show rapid convergence and stable configurations. Our combination of artificial forces and distributed algorithms produces a high-quality hexagonal lattice efficiently.

## 9. REFERENCES


[1] O. Yakimenko B. Eikenberry and M. Romano. A vision based navigation among multiple flocking robots: Modeling and simulation. *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2006.

[2] T. Balch and M. Hybinette. Social potentials for scalable multi-robot formations. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, 2000.

[3] J. Desai, J. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, December 2001.

[4] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[5] D. Gordon-Spears and W. Spears. Analysis of a phase transition in a physics-based multiagent system. *Lecture Notes in Computer Science*, page 193?207, 2003.

[6] Y. Hanada, Geunho Lee, and Nak Young Chong. Adaptive flocking of a swarm of robots based on local interactions. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 340 –347, april 2007.

[7] Eric Martinson and David Payton. Lattice formation in mobile autonomous sensor arrays. In Erol Sahin and William M. Spears, editors, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 98–111. Springer Berlin / Heidelberg, 2005. 10.1007.

[8] Sarah Barman Paolo Remagnino Robert J. Mullen, Dorothy Monekosso. Reactive coordination and adaptive lattice formation in mobile robotic surveillance swarms. *Distributed Autonomous Robotic*, 2010.

[9] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003.

[10] W. M. Spears, D. F. Spears, J. C. Hamann, and R. Heil. Distributed, Physics-Based control of swarms of vehicles. *Autonomous Robots*, 17(2):137–162, 2004.

[11] W. M. Spears, D. F. Spears, R. Heil, W. Kerr, and S. Hettiarachchi. An overview of physicomimetics. *Lecture Notes in Computer Science-State of the Art Series*, 3342, 2005.

[12] W.M. Spears, W.M. Spears, R. Heil, R. Heil, D.F. Spears, and D. Zarzhitsky. Physicomimetics for mobile robot formations. In *Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on*, pages 1528–1529, 2004.