Worst-case time complexity of a lattice formation problem

Ketan Savla and Francesco Bullo
Center for Control, Dynamical Systems and Computation
University of California at Santa Barbara
2338 Engineering Bldg II, Santa Barbara, CA 93106-5070
ketansavla@umail.ucsb.edu, bullo@engineering.ucsb.edu

***Abstract*— We consider a formation control problem for a robotic network with limited communication and controlled motion abilities. We propose a novel control structure that organizes the robots in concentric layers and that associates to each layer a local leader. Through a load balancing algorithm on the asynchronous network of layers we allocate the desired number of robots on each layer. A final uniform spreading algorithm leads the robots to a lattice-like formation. This novel distributed communication and control algorithm runs in linear time in the worst case.**

I. Introduction

Models for robotic networks are the subject of the early work in [1], where some impossibility results are established for certain formation control problems. More recently, Martínez et al in [2] propose a detailed model and analyze the time complexity of basic rendezvous and deployment algorithms. For many of the resulting linear dynamical systems, the worst case time complexity is of order $\Theta(n^2 \log n)$ (rendezvous) and $O(n^3 \log n)$ (deployment). Sharma et al [3] characterize the time complexity of sensor-based vehicle routing in 2 dimensions, i.e., of a distributed point to point motion planning problem. In their formulation, size and speed of the robots decrease with an increasing size of the network; in summary the worst case complexity of the vehicle routing problem is $\Theta(n)$ and the expected time complexity if $\Theta(\sqrt{n})$. Smith and Bullo in [4] consider a target assignment problem and characterize its time complexity as being $\Theta(n)$ in the worst case. It is not completely fair to compare all these results as the models they adopt have small but important differences. Additional related work includes the study of scaling laws in statistical mechanics, percolation theory, and wireless communication; e.g., see [5], [6], [7], [8].

The contributions of this paper are twofold. We consider a formation control problem for a robotic network with appropriate scaling laws inspired by the thermodynamic limit in statistical mechanics and by the memory and communication rate restrictions typical in distributed algorithms. Our formation control problem amounts to the problem of steering robots to "contiguous" vertices of a lattice formation in the plane. Our first contribution is to show that this problem has worst case time complexity of order $O(n)$. The second contribution is the explicit design of an communication and control law that achieves this upper bound. Our novel approach is based on the distributed computation and representation of a control structure called "rotating layers." The key idea is to organize the robots in concentric layers and to orchestrate their motion in such a way that the resulting network is connected over certain intervals. The main algorithmic contribution is a load balancing algorithm for an asynchronous chain network with bounded time complexity.

The paper is organized as follows. Section II presents the formal model with a particular attention to appropriate scaling laws and complexity notions. Section III introduces our novel control structure and establishes its properties. Section IV contain the main algorithmic contribution, namely a simple load balancing algorithm and its analysis. Finally, Section V describes simple strategies on how to place the robots in their desired positions inside the control structure.

II. Modeling the robotic network

Let n be the number of robots in the network, each with a unique identifier (UID). Let $I = \{1, \dots, n\}$ be the set of unique identifiers. We assume continuous-time motion and communication at discrete times. Each robot has the following sensing, computation, communication, and motion control capabilities. The memory and transmission capability of the processor is $O(\log(n))$ bits. The ith robot occupies a location p_i in a compact region $\mathcal{Q} \subset \mathbb{R}^2$, and it moves according to the continuous-time control system

$$\dot{p}_i = u_i,$$

with $|u| \leq u_{\max}$. We shall let the robots to always move at the maximum speed $u_{\max}$. The sensing and communication model is the following. The processor of each robot can sense its position with $1/\sqrt{n}$ accuracy. Consequently, each robot can represent its position with $O(\log(n))$ bits. Every robot is capable of transmitting its position to any other robot within a closed disk of radius $r_{\text{comm}} \in \mathbb{R}_+$. We assume that r_{comm} and $u_{\max}$ scales as $1/\sqrt{n}$ because of congestion. The inspiration for this scaling comes from the thermodynamic limit in statistical mechanics.

The n robots are initially randomly placed in $\mathcal{Q}$. The communication graph is the r_{comm}-disk graph, i.e., the graph $\mathcal{G}_{\text{disk}}$ with vertices $p_1, \ldots, p_n$ and edges (p_i, p_j) for all $i \neq j$ if and only if $\|p_i - p_j\| \leq r_{\text{comm}}$. We assume that the initial placement of the robots $p_1(0), \ldots, p_n(0)$ is such that their communication graph $\mathcal{G}_{\text{disk}}(p_1(0), \ldots, p_n(0)$ is connected.

The coordination objective is as follows: place the robots at "contiguous" locations in a hexagonal lattice with desired inter-robot distance $r_{\text{des}} \in \mathbb{R}_+$ (refer Fig 1), where r_{des} scales as $1/\sqrt{n}$ because of congestion. We assume that $r_{\text{des}} < r_{\text{comm}}/2$.

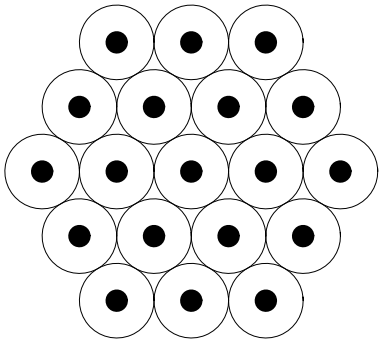

Fig. 1. Hexagonal lattice formation.

III. A CONTROL STRUCTURE BASED ON ROTATING LAYERS

In this section we outline the details of a control structure that the robots compute as part of their initialization before executing the main algorithm. The communication graph used for this initialization process is the r_{comm}-disk graph, i.e., two robots are able to communicate to each other only when they are within r_{comm} distance of each other. The control structure is constructed via a few intermediate steps.

Step 1: Central leader election

The robots estimate the boundaries of the *bounding box*, i.e., the smallest rectangle that is aligned with the coordinate axes and that contains all the robots. The robots then calculate their respective distances from the geometric center of this rectangle and the robot ℓ_0 that is closest to the center is elected as the leader. A flooding protocol to compute the bounding box is discussed in [9]; it is immediate to see that its time complexity is $\Theta(n)$, where n is the number of robots. As is well know [10], [11], the time complexity of leader election based on a flooding algorithm is also in $\Theta(n)$.

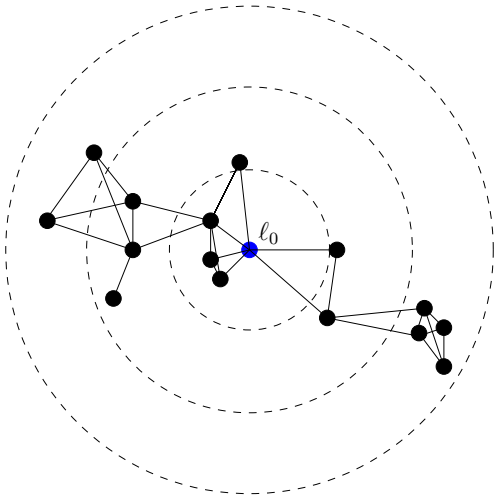

Fig. 2. Establishing layers

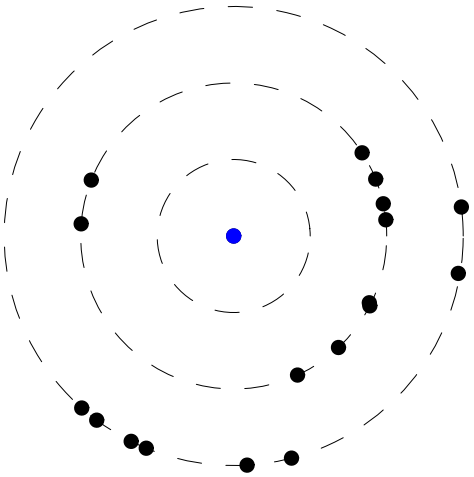

Fig. 3. Rotating Layers.

Step 2: Rotating layer formation

The second step in computing the control structure is the formation of rotating layers centered at the leader ℓ_0. To do this, the robots first determine the layer to which they belong based on their distances from the

leader ℓ_0 (refer Figure 2). The layer number of robot i, denoted by $L(i)$, is given by

$$L(i) = \left\lceil \frac{\|p_i - p_{\ell_0}\|}{r_{\text{des}}} \right\rceil.$$

Because the environment is bounded, so is the maximum layer number, that we denote with N_{layer}. As a consequence of the connectivity assumption and of the relationship between r_{comm} and r_{des}, one can see that if the annulus with radii $[jr_{\text{des}}, (j+1)r_{\text{des}}]$ is empty, then so are all other annuli with larger radii.

The robots then move radially away from p_{ℓ_0} to a distance from p_{ℓ_0} determined by their layer number. The nominal distance from ℓ_0 for robot i is $r_{\text{nom}}(i)$ defined by

$$r_{\text{nom}}(i) = L(i) r_{\text{des}}.$$

Once the robots reach their nominal distances as shown in Figure 3, the ones in the odd numbered layers rotate in the clockwise direction and the ones in the even numbered layers rotate in the counterclockwise direction. We shall refer to such layers as *rotating layers* (refer Figure 3). Without loss of generality, we assume that the speed $u_{\max}$ is sufficiently small, as compared with r_{comm}, so that any robot in layer number k is within communication range of all robots in layer numbers $k-1$ and $k+1$ at least once every $2\pi k r_{\text{des}}/u_{\max}$ time intervals. The resulting communication graph over an appropriate time interval is shown in Figure 4. This step takes $O(1)$ time.

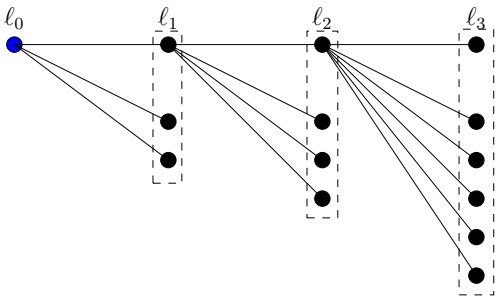

Fig. 4. Resulting "over-time" communication graph

Step 3: Layer leader elections

After all robots reach their nominal distances and begin rotating, the following protocol is executed. The objective is to elect a leader ℓ_j for each rotating layer j and to count the number of robots present in each layer.

1: set $j := 1$
2: robot ℓ_{j-1} counts the number of robots in layer j, denoted by $n_{\text{layer}}(j)$, during the next $2\pi j r_{\text{des}}/u_{\max}$ time interval
3: **if** $n_{\text{layer}}(j) = 0$ **then**
4: exit
5: **else**
6: among all the robots present in layer j, robot ℓ_{j-1} elects the one with the highest UID as ℓ_j and communicates $n_{\text{layer}}(j)$ to it
7: **end if**
8: Set $j := j + 1$ and go back to `2:`

For layer j, the counting and leader election process requires at most $4\pi j r_{\text{des}}/u_{\max}$ time. The following lemma characterizes the time complexity of the process required for the layer leader election step.

Lemma 3.1: Starting from the rotating layers, the time required to elect layer leaders belongs to $O(n)$.

Proof: Circumference of the jth layer is $2\pi j r_{\text{des}}(n) \sim j/\sqrt{n}$. Robot speed is $u_{\max}(n) \sim 1/\sqrt{n}$. Hence, steps `2:` and `6:` require $O(j)$. But the number of layers, $N_{\text{layer}} \in O(\sqrt{n})$. Therefore the total time required to elect layer leaders is $\sum_{j=1}^{\sqrt{n}} j \in O(n)$. ■

We are now ready to state the time complexity required to form the rotating layer structure.

Lemma 3.2: The time required to form the rotating layer structure from an initially connected set of n robots belongs to $O(n)$.

Analysis of communication across and inside layers

As the control structure is in place and robots rotate according to their layer identifier, the communication graph across robots changes with time. In other words, if the robot trajectories are denoted by $(p_1(t), \ldots, p_n(t))$, then the time-dependent communication graph is $t \mapsto \mathcal{G}_{\text{disk}}(p_1(t), \ldots, p_n(t))$. Because the number of layers belongs to $O(\sqrt{n})$, this time-dependent graph has the following useful properties (refer Figure 5):

(i) for all $j \geq 0$, there is an edge between ℓ_j and any robot in layer $j+1$ at least once every $2\pi j r_{\text{des}}/u_{\max} \in O(\sqrt{n})$ time instants, and
(ii) each leader ℓ_j can broadcast a message to all robots in its own layer j in time $4\pi j r_{\text{des}}/u_{\max} \in O(\sqrt{n})$.

IV. Layer balancing

The hexagonal lattice can be viewed as a layer formation around a central point with $6j$ points in layer number j. In order to arrange the robots into such

Fig. 5. Asynchronous chain network

a configuration, we first need to distribute the robots among the layers so that we have $6j$ robots in the j^{th} layer. We want this property to hold for as many consecutive layers as possible starting with layer 1. We call this the *layer balancing problem*. To solve this problem, we introduce a useful finite state machine Σ and design a load balancing algorithm for Σ. We later show how to implement this algorithm on the robotic network.

The state of the finite state machine Σ is the number of robots in each layer. Let N^* be the smallest positive integer such that $N^*(N^*+1) \geq n/3$ (this will be the final number of balanced layers). The number of states of Σ, denoted by N_{dyn}, is $\max\{N_{\text{layer}}, N^*\}$. Hence, the state of Σ can be described by the tuple $x(t) = (x_1(t), \ldots, x_{N_{\text{dyn}}}(t))$, where $x_j(0) = n_{\text{layer}}(j) \quad \forall 1 \leq j \leq N_{\text{layer}}$ and $x_j(0) = 0 \quad \forall N_{\text{layer}}+1 \leq j \leq N_{\text{dyn}}$. In addition to this, we define *auxiliary* state variables $y(t) = (y_1(t), \ldots, y_{N_{\text{dyn}}}(t))$ and $z(t) = (z_1(t), \ldots, z_{N_{\text{dyn}}}(t))$. These variables are initialized as follows: $y_1(0) = 1$, $y_j(0) = 0 \quad \forall i > 1$ and $z_j(0) = 0 \quad \forall i \geq 1$. These auxiliary variables $y(t)$ and $z(t)$ take values in $\{0, 1\}$ and their values influence the evolution of Σ. The evolution of Σ obeys the following discrete-time linear system:

$$x(t+1) = x(t) + \begin{bmatrix} +1 \\ -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u_1 + \ldots \begin{bmatrix} 0 \\ \vdots \\ 0 \\ +1 \\ -1 \end{bmatrix} u_{N_{\text{dyn}}-1},$$

where $u_j \in \mathbb{Z}$ is a control signal establishing how many robots should transfer from layer $j+1$ to j. For the system Σ, the layer balancing problem amounts to the selection of an input trajectory that steers $x(t)$ from the initial state $(x_1(0), \ldots, x_{N_{\text{dyn}}}(0))$ to a final state $(x_1(T), \ldots, x_{N_{\text{dyn}}}(T))$, where $x_j(T) = 6j \quad \forall 1 \leq j \leq N^*-1$, $x_{N^*} = n - \sum_{j=1}^{N^*} x_j(T)$ and $x_j(T) = 0 \quad \forall N^*+1 \leq j \leq N_{\text{dyn}}$ for some $T > 0$. This is akin to a load balancing problem on an asynchronous chain network.

We assume that only one control signal can be nonvanishing at each instant of time. In the following algorithm we denote by $\Sigma \leftarrow (\alpha, \beta)$ the transition of Σ generated by the control signal $u_\alpha = \beta$ with all other inputs equal to zero.

INCREMENTAL BALANCING ALGORITHM

for $j := 1$ to $(N_{\text{dyn}} - 1)$ **do**
  if $(y_j = 1)$ AND $(x_j \geq 6j)$ **then**
    {push excess robots to next layer}
    $\Sigma \leftarrow (j, 6j - x_j)$
    {relinquish the token to next leader}
    set $y_{j+1} := 1$, $y_j := 0$ and $z_j := 1$
  end if
  if $(y_j = 1)$ AND $(x_j > 6j)$ **then**
    {push excess robots to previous layer}
    $\Sigma \leftarrow (j-1, x_j - 6j)$
  end if

Lemma 4.1: The incremental balancing algorithm solves the layer balancing problem in time $O(n)$ in the worst case.

Proof: We only provide a sketch of the proof here and refer the reader to a forthcoming manuscript. If no layer leader ever waits, then leader j requires time of order $O(j)$ to execute the algorithm and the time complexity is $\sum_{j=1}^{O(\sqrt{n})} j = O(n)$. On the other hand, if the layer leader i waits for robots to arrive from the layer j with $j > i$, then one can deduce that the waiting time is at most $O(\sum_{k=i}^{j} k)$ and that, after i sets its z_i variable to 1, then no more layer leader will wait until layer leader j at least. In summary, the total waiting time is at most $O(\sum_{j=1}^{\sqrt{n}} j) = O(n)$. ■

Next, we explain how the robotic network equipped with the rotating layer structure can implement the INCREMENTAL BALANCING ALGORITHM in a sequential asynchronous manner.

We begin by designing some communication messages between layer leaders that encode various inter-layer cooperation requests. Based on the following requests one can easily envision how a distributed algorithm for the synchronous robotic network can be designed to exactly implement the INCREMENTAL BALANCING ALGORITHM for the asynchronous chained network.

Relinquish from j to $j+1$: Provided layer $j+1$ exists, leader ℓ_j communicates to ℓ_{j+1} that it is now its turn to evaluate if the layer $j+1$ is balanced or not.

Push excess robots from j to $j+1$: Provided layer j contains at least $6j$ robots, leader ℓ_j requests the excess robots in its layer to move to layer $j+1$ (possibly

creating it and subsequently electing a leader). The counters on layers j and $j+1$ are updated through a communication between ℓ_j and ℓ_{j+1}.

Push excess robots from j to $j-1$: Provided layer j contains at least $6j$ robots, leader ℓ_j requests the excess robots in its layer to move to layer $j-1$. The counters on layers j and $j-1$ are updated through a communication between ℓ_j and ℓ_{j-1}.

V. Uniform spreading

This phase is executed sequentially starting from innermost layer and progressing outwards. We provide an informal description in three steps:

(i) When the leader ℓ_j is in charge of the uniform spreading of the layer $j+1$, starting from ℓ_{j+1}, it does the following two things as it encounters a robot i in layer $j+1$. First, it determines the order number of the robot i in the layer, denoted by order(i), i.e., 1 for the leader ℓ_{j+1}, 2 for the next robot and so on. Second, it transmits the STOP signal and the quantity $\theta(i) = 2\pi(i-1)/n_{\text{layer}}(j+1)$ to the robot i.

(ii) When a robot i in layer $j+1$ receives these messages from the leader ℓ_j, it makes one full rotation and then stops at a position which extends an angle $\theta(i)$ at ℓ_0 with respect to the north direction. (One way to do this is storing coordinates of one point which is exactly north of ℓ_0).

(iii) After stopping, ℓ_{j+1} takes charge of the uniform distribution of the layer $j+2$. All the other robots in the layer $j+1$, move inwards by an *appropriate* distance until they are at a vertex on the hexagonal lattice. (One way to do this is by making every robot move radially inwards until it cannot move further without being closer than r_{des} to any other robot).

On the completion of this uniform spreading step, all the robots have occupied the vertices of a regular lattice. This step takes $O(n)$ time.

VI. Summary

In summary, we can now state a result for the total time required to solve the lattice formation problem using the procedure in this paper.

Theorem 6.1: The time required to solve the lattice formation problem for a robotic network described in Section II belongs to $O(n)$. This bound is achieved by the sequential computation of the rotating layers and execution of the incremental balancing and uniform spreading algorithms.

In future work we envision studying worst-case lower bounds as well as investigating stochastic scenarios in which the initial robot positions are randomly uniformly placed.

Acknowledgments

This material is based upon work supported in part by ARO MURI Award W911NF-05-1-0219. The second author would like to thank Stephen L. Smith for a number of useful discussions about congestion modeling and control structures in coordination problems.

References

[1] I. Suzuki and M. Yamashita, "Distributed anonymous mobile robots: Formation of geometric patterns," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1347–1363, 1999.

[2] S. Martínez, F. Bullo, J. Cortés, and E. Frazzoli, "On synchronous robotic networks – Part I: Models, tasks and complexity notions. & Part II: Time complexity of rendezvous and deployment algorithms," *IEEE Transactions on Automatic Control*, Apr. 2005. Submitted.

[3] V. Sharma, M. Savchenko, E. Frazzoli, and P. Voulgaris, "Time complexity of sensor-based vehicle routing," in *Robotics: Science and Systems* (S. Thrun, G. Sukhatme, S. Schaal, and O. Brock, eds.), pp. 297–304, Cambridge, MA: MIT Press, 2005.

[4] S. L. Smith and F. Bullo, "Target assignment for robotic networks: asymptotic performance under limited communication," in *American Control Conference*, (New York), July 2007. Submitted.

[5] M. Penrose, *Random Geometric Graphs*. Oxford Studies in Probability, Oxford, UK: Oxford University Press, 2003.

[6] L. Booth, J. Bruck, M. Franceschetti, and R. Meester, "Covering algorithms, continuum percolation and the geometry of wireless networks," *The Annals of Applied Probability*, vol. 13, no. 2, pp. 722–741, 2003.

[7] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[8] J. Urrutia, "Local solutions for global problems in wireless networks." Preprint, May 2006.

[9] G. Notarstefano and F. Bullo, "Distributed consensus on enclosing shapes and minimum time rendezvous," in *IEEE Conf. on Decision and Control*, (San Diego, CA), Dec. 2006. To appear.

[10] N. A. Lynch, *Distributed Algorithms*. San Mateo, CA: Morgan Kaufmann Publishers, 1997.

[11] D. Peleg, *Distributed Computing. A Locality-Sensitive Approach*. Monographs on Discrete Mathematics and Applications, Philadelphia, PA: SIAM, 2000. 0898714648.