Safe graph rearrangements for distributed connectivity of robotic networks

Michael Schuresko and Jorge Cortés

***Abstract*— This paper studies distributed algorithms for performing graph rearrangements that preserve the connectivity of a robotic network. Given a connected graph describing the topology of the network, preserving a fixed set of edges while performing a coordination task guarantees that connectivity is maintained. However, the preservation of a fixed set of edges often results in suboptimal and over-constrained robot operation. This paper presents a distributed algorithm to perform graph rearrangements that allow the robotic network to transform its interconnection topology between any two trees. We present a method for composing this algorithm with other control algorithms, and make preference guarantees about the choices of links to be preserved under the resulting composition. We use these ideas to propose a distributed formation morphing algorithm, and characterize its time complexity.**

I. Introduction

This paper considers the problem of maintaining connectivity of a robotic network while performing a coordination task. Given a group of robots and an interaction graph induced by the set of robot positions, we identify the following connectivity-related problems:

(i) How should the robots move so as to maximize some desired measure of connectivity subject to some position constraints?

(ii) Given a measure of the connectivity of the interaction graph, a connectivity threshold, and some coordination task, how should robots move to achieve the task subject to the value of the measure of connectivity never crossing the threshold?

We are motivated by the case in which the edge weights of the graph represent some measure of inter-robot communication channel capabilities.

Literature review: In [1], convex optimization is used to solve problem (i) in the presence of convex constraints on the space of edge weights. A solution to (i) with non-convex constraints is presented in [2]. An extension of similar methods to provide a distributed algorithm for (i) is presented in [3]. A commonly addressed sub-problem of (ii) occurs when one is merely concerned with whether the graph induced by the non-zero-weight edges of the interaction graph is 1-connected. The solution to this problem proposed in [4] allows for a general range of agent motions, but is not distributed. A distributed solution to this sub-problem appears in [5], but the solution as presented requires that the robots maintain a fixed set of edges. If the configurations the formation is to switch between are known in advance, then [6] provides a robust method for transitioning between them. Many solutions exist which satisfy non-zero-edge 1-connectivity for a specific task, among the most notable is [7], in which connectivity-preserving motions are generated between pairs of formations. To our knowledge, there are no solutions, distributed or otherwise, to problem (ii), beyond those described above. Much of the related literature deals with algorithms to repair a spanning tree after link failure, rather than adjusting a spanning tree to allow agents to break desired links. For a survey of such algorithms see [8] and references contained therein. Finally, our technical approach uses the modeling framework proposed in [9] to combine connectivity maintenance algorithms with other control algorithms.

Statement of contributions: This paper introduces the Connectivity Maintenance Algorithm for dynamically agreeing upon a subgraph (specifically, a tree) of a proximity graph. Maintaining each edge of the tree maintains connectivity of the robotic network. The advantage of the proposed algorithm is that it allows for on-line topological rearrangements of the tree in a distributed manner. We analyze the correctness of this algorithm, and show that the allowed rearrangements are sufficient to allow configuration changes between any two constraint trees. The paper also introduces the notion of *input-output control and communication law*. One can formally compose different input-output control and communication laws to yield a coordination algorithm whose execution can be characterized by studying the individual components. Our Connectivity Maintenance Algorithm is an example of an input-output control and communication law, which, in Section IV, is composed with another law to synthesize a formation morphing algorithm. We show that given initial and final configurations, the formation morphing algorithm steers the robotic network from one to the other while maintaining connectivity in the r-disk proximity graph. We also characterize the time complexity of the algorithm, and present simulations that confirm the theoretical analysis.

Organization: Section II introduces useful notions from graph theory, proximity graphs, and the robotic network model. Section III presents, and analyzes the Connectivity Maintenance Algorithm algorithm. Section IV introduces the formation morphing algorithm, studies its correctness, and analyzes its time complexity. Section V presents simulations confirming our results and Section VI presents our conclusions.

Notation: Throughout the paper, $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{R}_{>0}$ denote the sets of reals, non-negative reals, and positive reals, respectively. For a set S, $\mathbb{F}(S)$ denotes the collection of all finite subsets of S. Whenever we provide algorithm pseudo-

Michael Schuresko is with the Department of Applied Mathematics and Statistics, University of California, Santa Cruz, CA, USA, mds@soe.ucsc.edu

Jorge Cortés is with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA, USA, cortes@ucsd.edu

code, we use $a \leftarrow b$ to mean "a is assigned a value of b." We use $f(n) \in O(g(n))$ to mean that there exist $N_0 \in \mathbb{N}$, $c \in \mathbb{R}$ such that $f(n) < cg(n)$ for all $n > N_0$; we use $f(n) \in \Omega(g(n))$ to mean that there exist $N_0 \in \mathbb{N}$, $c \in \mathbb{R}$ such that $f(n) > cg(n)$ for all $n > N_0$. Finally, $f(n) \in \Theta(g(n))$ means $f(n) \in O(g(n)) \cap \Omega(g(n))$. We use $\wedge$ to mean "logical and" and $\vee$ to mean "logical or."

II. Preliminary developments

In this section, we review some useful notions from graph theory and computational geometry. We also introduce a formal model for robotic networks and coordination algorithms.

A. Graph-theoretic notions

Here, we recall some standard notions from graph theory [10], [11], [12]. A directed graph, or *digraph*, is a pair of sets, $G = (V, E)$, such that $E \subseteq V \times V$. Elements of V and E are known as vertices and edges, respectively. An undirected graph, or simply *graph*, $G = (V, E)$, consists of a vertex set V and a set E of unordered pairs of vertices. Given a digraph (V, E), one can define the associated underlying undirected graph (V, E') by setting $(u, v) \in E$ implies $(u, v), (v, u) \in E'$. A digraph (V', E') is a *subgraph* of a digraph (V, E) if $V' \subset V$ and $E' \subset E$; additionally, a digraph (V', E') is a *spanning* subgraph if it is a subgraph and $V' = V$. Two digraphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there exists a bijective function f mapping V_1 onto V_2 such that $(i, j) \in E_1$ if and only if $(f(i), f(j)) \in E_2$. From this point on, for a graph on n nodes (i.e., $|V| = n$) we assume without loss of generality that $V = \mathbb{Z}_n = \{0, \dots, n-1\}$, thus allowing us to refer to "node 0," etc. Given a graph, $G = (V, E)$, the set of neighbors of node $i \in V$ is $\mathcal{N}(i) = \{j \in V \mid (i, j) \in E \vee (j, i) \in E\}$.

A *tree* is a connected graph with no cycles. A *directed tree* T is a digraph whose underlying undirected graph is a tree. In this paper, we only deal with directed rooted trees. In a rooted tree, each edge connects a *child* node i node to its *parent* node $p^{[i]}_{\text{curr}}$. The unique node with no parents is called the *root*, and the distance in a tree from a node i to the root is called the *depth* of i, denoted $\text{dp}^{[i]}_T$. Nodes i and j are called *siblings* in a given tree if they have the same parent, $p^{[i]}_{\text{curr}} = p^{[j]}_{\text{curr}}$. We say i is a *descendant* of j, or equivalently j is an *ancestor* of i, if there exists a sequence of nodes, $k_1, \cdots, k_n$ such that $p^{[i]}_{\text{curr}} = k_1, p^{[k_1]}_{\text{curr}} = k_2, \dots, p^{[k_n]}_{\text{curr}} = j$.

B. Proximity graphs

We use proximity graphs as an abstraction of network interconnection among spatially distributed agents. Proximity graphs associate network topology with robot positions by defining mappings from finite collections of points in $\mathbb{R}^d$ to graphs, e.g., see [13], [14]. For $\mathcal{P} \in \mathbb{F}(\mathbb{R}^d)$, let $\mathbb{G}(\mathcal{P})$ denote the set of undirected graphs whose vertex set is some labeling of the elements in $\mathcal{P}$. A *proximity graph* $\mathcal{G} : \mathbb{F}(\mathbb{R}^d) \to \mathbb{G}(\mathbb{F}(\mathbb{R}^d))$ associates to $\mathcal{P} \in \mathbb{F}(\mathbb{R}^d)$, $|\mathcal{P}| = n$, an undirected graph in $\mathbb{G}(\mathcal{P})$ with vertex set isomorphic to $\mathbb{Z}_n$ and edge set $\mathcal{E}_{\mathcal{G}}(\mathcal{P})$, where $\mathcal{E}_{\mathcal{G}} : \mathbb{F}(\mathbb{R}^d) \to \mathbb{F}(\mathbb{Z}_n \times \mathbb{Z}_n)$. For convenience, let $i_{\mathbb{F}} : (\mathbb{R}^d)^n \mapsto \mathbb{F}(\mathbb{R}^d)$ take a tuple $P = (p_1, \dots, p_n)$ of n points in $\mathbb{R}^d$ to the finite collection $\mathcal{P} = \{p_1, \dots, p_n\} \in \mathbb{F}(\mathbb{R}^d)$. Thus, given a tuple of agent positions, P, and a proximity graph $\mathcal{G}$, $\mathcal{G}(i_{\mathbb{F}}(P))$ is the graph induced by P under $\mathcal{G}$.

Definition 2.1 (Configuration): Given a proximity graph $\mathcal{G}$, let (T, P) be a pair consisting of a tree T on n nodes and a vector $P \in (\mathbb{R}^d)^n$ of agent positions such that T is a spanning subgraph of $\mathcal{G}(i_{\mathbb{F}}(P))$. A *$\mathcal{G}$-configuration* $[(T, P)]$ of n agents in d-dimensional space is the equivalence class of (T, P) under the relation $\cong_{\text{rot}}$ defined by $(T_1, P_1) \cong_{\text{rot}} (T_2, P_2)$ if $T_1 = T_2$ and there is some bijective affine transformation that maps P_1 onto P_2.

When the proximity graph in question is clear, we simply use the word "configuration." Note that $\mathcal{G}$-configurations exist only if $\mathcal{G}(i_{\mathbb{F}}(P))$ is connected. We use $[(T, P)]$ to denote the equivalence class containing (T, P), and refer to T as the "constraint tree."

C. Robotic network model

We present our algorithms within the framework introduced in [9] for synchronous robotic networks. For completeness, we present a brief account of the model here.

Definition 2.2 (Robotic network): A *robotic network* $\mathcal{S}$ is a tuple $(I, \mathcal{A}, E_{\text{cmm}})$ consisting of

(i) $I = \{0, \dots, n-1\}$; the *set of unique identifiers (UIDs)*;
(ii) $\mathcal{A} = \{A^{[i]}\}_{i \in I}$, with $A^{[i]} = (X^{[i]}, U^{[i]}, X^{[i]}_0, f)$, the *set of physical agents*; here $X^{[i]}$ is the state-space of the ith control system and $U^{[i]}$ is the control space of the ith control system;
(iii) E_{cmm}, the *communication edge map*, is a map from $\prod_{i \in I} X^{[i]}$ to the subsets of $I \times I \setminus \text{diag}(I \times I)$.

If $A^{[i]} = (X, U, X_0, f)$ for all $i \in I$, then the robotic network is called *uniform*.

Next we introduce the notion of input-output control and communication law. This notion is a generalization of the concept of *control and communications law* proposed in [9], and aims at facilitating the composition of reusable algorithmic components.

Definition 2.3: A *(synchronous, static, uniform, feedback) input-output control and communication law* $\mathcal{CC}$ for a uniform network $\mathcal{S}$ consists of the sets:

(i) $\mathbb{T} = \{t_\ell\}_{\ell \in \mathbb{N}_0} \subset \mathbb{R}_{\geq 0}$, a communication schedule;
(ii) L, a communication language;
(iii) $W^{[i]} = W$, $i \in I$, sets of values of *logic variables* $w^{[i]}$, $i \in I$;
(iv) $W_0^{[i]} \subseteq W$, $i \in I$, subsets of *allowable initial values*;
(v) ${W_{\text{in}}}^{[i]} = W_{\text{in}}$, sets of values of *input logic variables* ${w_{\text{in}}}^{[i]}$, $i \in I$;
(vi) ${W_{\text{in}}}_0^{[i]} \subseteq W_{\text{in}}$, subsets of *allowable initial input values*;
(vii) ${W_{\text{out}}}^{[i]} = W_{\text{out}}$, sets of values of *output logic variables* ${w_{\text{out}}}^{[i]}$, $i \in I$;

and of the maps:

(i) $\text{msg} : \mathbb{T} \times X \times W \times W_{\text{in}} \times I \to L$, the *message-generation function*;
(ii) $\text{stf} : \mathbb{T} \times W_{\text{in}} \times W \times L^n \to W \times W_{\text{out}}$ the *(input-output) state-transition function*;
(iii) $\text{ctl} : \bar{\mathbb{R}}_+ \times X \times X \times W \times W_{\text{in}} \times L^n \to U$, $i \in I$, the *control function*.

For notational convenience, we often write an input-output state-transition function, stf as the pair $(\text{stf}_{\text{lv}}, \text{stf}_{\text{out}})$, where stf_{lv} computes values in W and stf_{out} in W_{out}. We will sometimes call stf_{out} the "output state transition function."

By a *control and communication law* we mean an input-output control and communication law with no inputs and no outputs, i.e., $W_{\text{in}} = \emptyset = W_{\text{out}}$. This definition is equivalent to the definition put forth in [9]. When we refer to an "evolution" of a robotic network, we mean the behavior of the network starting from a valid initial state. The execution of a control and communication law can be roughly described as follows: at each communication round, each agents sends messages to its neighbors according to the evaluation of msg. With the messages received, each agent updates the value of its logic variables using stf. In between communication rounds, the motion of each agent motion is governed by ctl. A precise description of an execution can be found in [9].

Definition 2.4: (Composition of input-output laws) The composition of two input-output control and communication laws $\mathcal{CC}_1$ and $\mathcal{CC}_2$, subject to $\mathcal{CC}_2 W_{\text{out}} = \mathcal{CC}_1 W_{\text{in}}$ and $\mathcal{CC}_1 W_{\text{out}} = \mathcal{CC}_2 W_{\text{in}}$, is a control and communications law, $\mathcal{CC}_1 \otimes \mathcal{CC}_2 = (\mathbb{T}, L, W, W_0, \text{msg}, \text{stf}, \text{ctl})$, with sets

$$\begin{aligned}
\mathbb{T} &= \mathcal{CC}_1\mathbb{T} = \mathcal{CC}_2\mathbb{T},\\
L &= \mathcal{CC}_1 L \times \mathcal{CC}_2 L,\\
W &= \mathcal{CC}_1 W \times \mathcal{CC}_2 W_{\text{in}} \times \mathcal{CC}_2 W \times \mathcal{CC}_1 W_{\text{in}},\\
W_0 &= \mathcal{CC}_1 W_0 \times \mathcal{CC}_2 W_{\text{in}0} \times \mathcal{CC}_2 W_0 \times \mathcal{CC}_1 W_{\text{in}0},
\end{aligned}$$

and functions

$$\begin{aligned}
\text{msg}(t, x, w) &= (\mathcal{CC}_1\text{msg}(t, x, \mathcal{CC}_1 w, \mathcal{CC}_1 w_{\text{in}}),\\
&\qquad \mathcal{CC}_2\text{msg}(t, x, \mathcal{CC}_2 w, \mathcal{CC}_2 w_{\text{in}})),\\
\text{stf}(t, w, l) &= (\mathcal{CC}_1\text{stf}(t, \mathcal{CC}_1 w_{\text{in}}, \mathcal{CC}_1 w),\\
&\qquad \mathcal{CC}_2\text{stf}(t, \mathcal{CC}_2 w_{\text{in}}, \mathcal{CC}_2 w)),\\
\text{ctl}(t, x_{t_\ell}, x, w^{[i]}) &= \mathcal{CC}_1\text{ctl}(t, x_{t_\ell}, x, \mathcal{CC}_1 w, \mathcal{CC}_1 w_{\text{in}})\\
&\qquad + \mathcal{CC}_2\text{ctl}(t, x_{t_\ell}, x, \mathcal{CC}_2 w, \mathcal{CC}_2 w_{\text{in}}).
\end{aligned}$$

In other words, the composition of two input-output control and communication laws is the natural result of substituting each law's output for the other law's input.

III. Connectivity Maintenance Algorithm

This section introduces the Connectivity Maintenance Algorithm. Section III-A describes the algorithm in detail and Section III-B analyzes its properties. The algorithm by itself does not invoke either physical agents or their mobility, and fits within common frameworks of distributed algorithms, see e.g., [15]. We present it as an input-output control and communication law as defined in Section II-C.

A. Algorithm description

Given a uniform network $\mathcal{S}$ with communication edge map determined by a proximity graph $\mathcal{G}$, the Connectivity Maintenance Algorithm is an input-output control and communication law $\mathcal{CC}$ for $\mathcal{S}$ consisting of the sets:

(i) $\mathbb{T} = \{t_\ell\}_{\ell \in \mathbb{N}_0} \subset \mathbb{R}_{\geq 0}$;
(ii) $L = W$;
(iii) $W = \mathbb{N}^4 \times \mathbb{Z}_2$, $i \in I$, are sets of values of the *logic variables* $w^{[i]} = (p_{\text{curr}}^{[i]}, \text{dp}_{\text{est}}^{[i]}, \text{phase}^{[i]}, p_{\text{next}}^{[i]}, \mathbb{I}_{\text{par-less}}^{[i]})$, $i \in I$, consisting of a parent identifier $p_{\text{curr}}^{[i]}$, a "depth estimate" $\text{dp}_{\text{est}}^{[i]}$, a round counter indicating the current mode of the algorithm $\text{phase}^{[i]}$, a proposed next parent $p_{\text{next}}^{[i]}$, and a boolean indicator $\mathbb{I}_{\text{par-less}}^{[i]}$ denoting whether i's parent $p_{\text{curr}}^{[i]}$ had a strictly smaller depth estimate than i as of the most recent communication round;
(iv) $W_0^{[i]} = \{(p_{\text{curr}}^{[i]}, \text{dp}_{\text{est}}^{[i]}, 0, p_{\text{curr}}^{[i]}, \text{false})\} \subseteq W$, $i \in I$ such that $p_{\text{curr}}^{[i]}$, $i \in I$, induces a connected tree T and $\text{dp}_{\text{est}}^{[i]}$ is the depth of i in T;
(v) W_{in}, are sets of values of *input logic variables*, ${w_{\text{in}}}^{[i]} : \mathbb{Z}_n \mapsto \mathbb{N} \cup \{\infty\}$ which specify a preference for attaching to one node over another. If ${w_{\text{in}}}^{[i]}(k) < {w_{\text{in}}}^{[i]}(j)$, then node i would prefer to attach to k over j. By convention, any domain element on which the action of the function is unspecified maps to ∞.
(vi) For simplicity, we let ${w_{\text{in}0}}^{[i]}$ map each $j \in \mathbb{Z}_n$ to ∞;
(vii) $W_{\text{out}} = \mathbb{Z}_n$ are sets of values of *output logic variables* ${w_{\text{out}}}^{[i]} = p_{\text{curr}}^{[i]}$ for $i \in I$;

and of the maps:

(i) **function** $\text{msg}(t, x, (p_{\text{curr}}^{[\text{id}]}, \text{dp}_{\text{est}}^{[\text{id}]}, \text{phase}^{[\text{id}]}), w_{\text{in}}, j) = (\text{id}, \text{dp}_{\text{est}}^{[\text{id}]}, p_{\text{curr}}^{[\text{id}]}, p_{\text{next}}^{[\text{id}]}, \mathbb{I}_{\text{par-less}}^{[\text{id}]})$, where id is the unique identifier of the sending agent;
(ii) **function** $\text{stf}_{\text{lv}}(t, w_{\text{in}}, (p_{\text{curr}}^{[\text{id}]}, \text{dp}_{\text{est}}^{[\text{id}]}, \text{phase}^{[\text{id}]}, p_{\text{next}}^{[\text{id}]}), l)$ as defined in Table I;

function $\text{stf}_{\text{lv}}(t, w_{\text{in}}, (p_{\text{curr}}^{[\text{id}]}, \text{dp}_{\text{est}}^{[\text{id}]}, \text{phase}^{[\text{id}]}, p_{\text{next}}^{[\text{id}]}), l)$

if $\text{phase}^{[\text{id}]} = 0$ **then**
  if $p_{\text{next}}^{[\text{id}]} \neq p_{\text{curr}}^{[p_{\text{next}}^{[\text{id}]}]}$ and $\text{dp}_{\text{est}}^{[p_{\text{next}}^{[\text{id}]}]} = \text{dp}_{\text{est}}^{[\text{id}]}$ and $\text{id} > \max_{\{j \mid p_{\text{next}}^{[j]} = \text{id} \vee j = p_{\text{next}}^{[\text{id}]}\}}(j)$ */*If my proposed parent is of my depth ($dp_{est}^{[p_{next}^{[id]}]} = dp_{est}^{[id]}$)and re-attaching ($p_{next}^{[p_{next}^{[id]}]} \neq p_{curr}^{[p_{next}^{[id]}]}$) and my unique id is greater then that of both my proposed parent, and any nodes that have proposed attaching to me as a parent, then discard my proposed parent and keep my current parent*/*
  then
    return $((p_{\text{curr}}^{[\text{id}]}, \text{dp}_{\text{est}}^{[\text{id}]}, (\text{phase}^{[\text{id}]} + 1) \bmod 4), p_{\text{curr}}^{[\text{id}]}, \mathbb{I}_{\text{par-less}}^{[\text{id}]})$
  else
    return $((p_{\text{next}}^{[\text{id}]}, \text{dp}_{\text{est}}^{[\text{id}]}, (\text{phase}^{[\text{id}]} + 1) \bmod 4), p_{\text{next}}^{[\text{id}]}, \mathbb{I}_{\text{par-less}}^{[\text{id}]})$
if $\text{phase}^{[\text{id}]} = 1$ **then**
  Set $\text{dp}_{\text{est}}^{[\text{id}]} \leftarrow \text{dp}_{\text{est}}^{[p_{\text{curr}}^{[\text{id}]}]} + 1$ */*Call this step the "update rule"*/*
  return $((p_{\text{curr}}^{[\text{id}]}, \text{dp}_{\text{est}}^{[\text{id}]}, (\text{phase}^{[\text{id}]} + 1) \bmod 4), p_{\text{next}}^{[\text{id}]}, \mathbb{I}_{\text{par-less}}^{[\text{id}]})$
if $\text{phase}^{[\text{id}]} = 2$ **then**
  Set $\mathbb{I}_{\text{par-less}}^{[\text{id}]} \leftarrow \text{false}$
  if $\text{dp}_{\text{est}}^{[p_{\text{curr}}^{[\text{id}]}]} < \text{dp}_{\text{est}}^{[\text{id}]}$ **then**
    Set $\mathbb{I}_{\text{par-less}}^{[\text{id}]} \leftarrow \text{true}$
  return $((p_{\text{curr}}^{[\text{id}]}, \text{dp}_{\text{est}}^{[\text{id}]}, (\text{phase}^{[\text{id}]} + 1) \bmod 4), p_{\text{next}}^{[\text{id}]}, \mathbb{I}_{\text{par-less}}^{[\text{id}]})$
if $\text{phase}^{[\text{id}]} = 3$ **then**
  Let $p_{\text{next}}^{[\text{id}]} \leftarrow \text{argmin}_{w_{\text{in}}(j)}\{j \in \mathcal{N}(\text{id}) | \text{dp}_{\text{est}}^{[j]} < \text{dp}_{\text{est}}^{[\text{id}]}$ or $p_{\text{curr}}^{[j]} = p_{\text{curr}}^{[\text{id}]}$ or $\text{dp}_{\text{est}}^{[j]} = \text{dp}_{\text{est}}^{[\text{id}]}$ and $\mathbb{I}_{\text{par-less}}^{[j]} = \text{true}\}$
  return $((p_{\text{curr}}^{[\text{id}]}, \text{dp}_{\text{est}}^{[\text{id}]}, (\text{phase}^{[\text{id}]} + 1) \bmod 4), p_{\text{next}}^{[\text{id}]}, \mathbb{I}_{\text{par-less}}^{[\text{id}]})$

TABLE I
stf_{lv} FOR THE CONNECTIVITY MAINTENANCE ALGORITHM.

(iii) **function** $\text{stf}_{\text{out}}(t, w_{\text{in}}, (p_{\text{curr}}^{[\text{id}]}, \text{dp}_{\text{est}}^{[\text{id}]}, \text{phase}^{[\text{id}]}, p_{\text{next}}^{[\text{id}]}), l) = p_{\text{curr}}^{[\text{id}]}$.

(iv) $\mathrm{ctl}(t, x_{t_\ell}, x, w, w_{\text{in}}) = 0$.

Remark 3.1 (Re-attach operations): The CONNECTIVITY MAINTENANCE ALGORITHM allows for two types of graph re-arrangements:

(i) The first type of re-arrangement is a *re-attach* of i to j having $\text{dp}_{\text{est}}^{[j]} < \text{dp}_{\text{est}}^{[i]}$. If agent i determines that it would rather have its parent be j rather than its current parent (via ${w_{\text{in}}}^{[i]}$), it first checks that $\text{dp}_{\text{est}}^{[j]} < \text{dp}_{\text{est}}^{[i]}$ (line 15 of Table I), notifies its current parent $p_{\text{curr}}^{[i]}$ and j of the proposed move on the notification step (via msg), and performs the topology rearrangement (line 3 of Table I) on the rearrangement step.

(ii) The second type of re-arrangement is a *re-attach* of i to j having $\text{dp}_{\text{est}}^{[j]} = \text{dp}_{\text{est}}^{[i]}$. We show later that no descendant of i has a lesser depth estimate then i, and that the rules concerning $\mathbb{I}_{\text{par-less}}^{[i]}$ prevent i from attaching to a descendant with equal depth estimate. The tie-breaking procedure of line 2 in Table I ensures that no cycles are created by simultaneous re-attach operations between agents of equal depth estimate. •

B. Correctness analysis and reachability

In this section, we analyze the CONNECTIVITY MAINTENANCE ALGORITHM. In particular, we show that connectivity is preserved throughout the execution of the algorithm, and that for any two trees, T_1 and T_2, there is a sequence of inputs that cause the algorithm to transform T_1 onto T_2. For convenience in the forthcoming analysis, we let $\text{rnd}(t) \in \mathbb{N}$ be the number of times the assignment $\text{phase}^{[i]} \leftarrow 2$ has been made at time t. We denote the value of, say $\text{dp}_{\text{est}}^{[i]}$ at iteration $\text{rnd}(t)$ by $\text{dp}_{\text{est}}^{[i]}(\text{rnd}(t))$.

Theorem 3.2: The execution of CONNECTIVITY MAINTENANCE ALGORITHM verifies that

- $\text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)) \leq \text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)-1) + 1$,
- $\text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)) \geq \text{dp}_{\text{est}}^{[p_{\text{curr}}^{[i]}]}(\text{rnd}(t))$,

for all $i \in \{0, \ldots, n-1\}$, where for convenience, $\text{dp}_{\text{est}}^{[i]}(r) = \text{dp}_{T}^{[i]}(t_0)$ for all rounds $r \leq 0$. Thus, at any time $t \geq 0$, if k is an ancestor of i, then $\text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)) \geq \text{dp}_{\text{est}}^{[k]}(\text{rnd}(t))$.

Proof: Note that 7 of CONNECTIVITY MAINTENANCE ALGORITHM is the only step where the value of $\text{dp}_{\text{est}}^{[i]}$ is modified. We refer to this step as the *update rule*. We induct on the current round, $\text{rnd}(t)$. Let j be $p_{\text{curr}}^{[i]}$ at iteration $\text{rnd}(t)$. This can only happen because either (i) j became i's parent due to a re-attach or (ii) j was i's parent at $\text{rnd}(t)-1$. In case (i), the re-attach requires $\text{dp}_{\text{est}}^{[j]}(\text{rnd}(t)-1) \leq \text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)-1)$, and this implies that $\text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)) \leq \text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)-1)+1$. In case (ii), $\text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)-1) = \text{dp}_{\text{est}}^{[j]}(\text{rnd}(t)-2)+1$. The induction hypothesis implies that $\text{dp}_{\text{est}}^{[j]}(\text{rnd}(t)-1) \leq \text{dp}_{\text{est}}^{[j]}(\text{rnd}(t)-2)+1$, and therefore $\text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)) \leq \text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)-1)+1$, thus proving the first item. Either i has attached to $p_{\text{curr}}^{[i]}$ more recently than $\text{dp}_{\text{est}}^{[p_{\text{curr}}^{[i]}]}$ has changed, or, by the update rule, $\text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)) \geq \text{dp}_{\text{est}}^{[p_{\text{curr}}^{[i]}]}(\text{rnd}(t)-1)+1$, from which we get $\text{dp}_{\text{est}}^{[i]}(\text{rnd}(t)) \geq \text{dp}_{\text{est}}^{[p_{\text{curr}}^{[i]}]}(\text{rnd}(t))$ which proves the second item. ■

Theorem 3.3: At all times during the execution of CONNECTIVITY MAINTENANCE ALGORITHM, the graph induced by the parent relation among the agents contains no cycles (and is therefore a connected tree).

Proof: Any cycle in the graph must consist entirely of agents having the same depth estimate (this follows from $\text{dp}_{\text{est}}^{[i]} \geq \text{dp}_{\text{est}}^{[p_{\text{curr}}^{[i]}]}$ from Theorem 3.2 and the fact that for a cycle, C, $\sum_{i \in C} \text{dp}_{\text{est}}^{[i]} - \text{dp}_{\text{est}}^{[p_{\text{curr}}^{[i]}]} = 0$). During the 4 communication rounds leading up to the creation of any cycle, all agents involved in the cycle must have the same depth estimate as well, otherwise there exists either an agent $i \in C$ having $\text{dp}_{\text{est}}^{[p_{\text{curr}}^{[i]}]} > \text{dp}_{\text{est}}^{[i]}$, which violates Theorem 3.2, or an agent $i \in C$ having $\text{dp}_{\text{est}}^{[p_{\text{next}}^{[i]}]} > \text{dp}_{\text{est}}^{[i]}$, which is prohibited by line 15 of CONNECTIVITY MAINTENANCE ALGORITHM.

When a cycle first appears, it must appear due to some agent, i, connecting to a new parent, j, with the same depth estimate. However, this new parent must, in turn, attach to an agent of the same depth estimate. It cannot have had a parent of the same depth estimate before the cycle was created, as that would have set $\mathbb{I}_{\text{par-less}}^{[j]} = \text{false}$ and prevented i from connecting to j. This argument can be carried all the way around the cycle, C, and we can conclude that for each agent $k \in C$, k attached to a new parent at the time the cycle was formed.

Here we invoke the UID-based tie-breaking scheme of line 2, and note that some agent, i, in this cycle must have been greater then either of its neighbors in C, which explicitly triggers 3, preventing i from attaching to $p_{\text{next}}^{[i]}$ and thus preventing the formation of a cycle. ■

Definition 3.4: An input-output control and communication law CC_c is *fully compatible with* CONNECTIVITY MAINTENANCE ALGORITHM if the following holds:

- W_{out} and W_{in} of CC_c match up, respectively, with W_{in} and W_{out} of CONNECTIVITY MAINTENANCE ALGORITHM;
- The time schedule, output state transition function (stf_{out}), and control function of CC_c are such that the control function is guaranteed never to induce a motion which causes $(i, p_{\text{next}}^{[i]})$ or $(i, p_{\text{curr}}^{[i]})$ to cease to be an edge of the underlying proximity graph.

Corollary 3.5: The composition of CONNECTIVITY MAINTENANCE ALGORITHM with a fully compatible input-output control and communication law, CC_c, is a control and communication law with the property that any robotic network which starts with a connected proximity graph remains connected throughout its execution.

Proof: By Theorem 3.3, the graph induced by edges of the form $(i, p_{\text{curr}}^{[i]})$ remains connected at all times. By the conditions imposed by the notion of "fully compatible with *fully compatible with* CONNECTIVITY MAINTENANCE ALGORITHM",

- each edge, $(i, p_{\text{curr}}^{[i]})$, remains in $\mathcal{G}$ at all times
- and any time the network switches $p_{\text{curr}}^{[i]}$ to $p_{\text{next}}^{[i]}$, it remains connected.

■

Next, we show that the trees produced by this algorithm are somehow better than an arbitrarily chosen tree.

Proposition 3.6: Suppose two nodes, i and j, are neighbors in the current communication graph at some time t_ℓ and $\mathbb{I}^{[i]}_{\text{par-less}} = \mathbb{I}^{[j]}_{\text{par-less}} = \text{true}$. Suppose further that ${w_{\text{in}}}^{[i]}(j) < {w_{\text{in}}}^{[i]}(k)$ for all $k \in \mathcal{N}(i)$ for $t_\ell \le t \le t_{\ell+4}$ and ${w_{\text{in}}}^{[j]}(i) < {w_{\text{in}}}^{[j]}(m)$ for all $m \in \mathcal{N}(j)$ for all $t_\ell \le t \le t_{\ell+4}$. Then if i and j remain neighbors in $\mathcal{G}$ for the next 4 communication rounds, either i will be $p^{[j]}_{\text{curr}}$, or j will be $p^{[i]}_{\text{curr}}$.

Proof: At some point during the next 4 rounds, i will set $p^{[i]}_{\text{next}} \leftarrow j$ and j will set $p^{[j]}_{\text{next}} \leftarrow i$ (line 15 of CONNECTIVITY MAINTENANCE ALGORITHM). If $\text{dp}^{[j]}_{\text{est}} < \text{dp}^{[i]}_{\text{est}}$ then i will attach to j since the tie-breaking scheme in lines 2 - 5 does not trigger unless $\text{dp}^{[j]}_{\text{est}} = \text{dp}^{[i]}_{\text{est}}$. If $\text{dp}^{[i]}_{\text{est}} = \text{dp}^{[j]}_{\text{est}}$ they will both propose to attach to each-other, in which case node with the smaller UID will attach to the node with the larger UID. ■

IV. FORMATION MORPHING PROBLEM

Here, we illustrate the utility of the CONNECTIVITY MAINTENANCE ALGORITHM introduced in Section III. We design a coordination algorithm that allows the network to move between any two different formations while maintaining connectivity. We begin by formally stating the problem.

Definition 4.1 (Formation morphing problem): Given a proximity graph, $\mathcal{G}$, the formation morphing problem is that of designing a distributed algorithm to compute motion between two configurations, $[(P_s, T_s)]$ and $[(P_{\text{targ}}, T_{\text{targ}})]$, of n robots in d-dimensional space such that the graph $\mathcal{G}(i_{\mathbb{F}}(P))$ remains connected at all times and the network reaches $[(P_{\text{targ}}, T_{\text{targ}})]$ in finite time.

Next, Section IV-A describes the algorithm in detail and Section IV-B analyzes its correctness.

A. Algorithm framework and specification

In this section, we introduce the FORMATION MORPHING ALGORITHM to solve the formation morphing problem. For clarity, we use the symbols $a_{i,1}, \ldots, a_{i,m_i}$, where m_i is the depth of i in the target constraint tree, T_{targ}, to denote the ancestors of i in T_{targ}. We use $a_{i,1}$ to mean the parent of i in T_{targ}, and a_{i,m_i} to mean the root of T_{targ}.

Consider a uniform network $\mathcal{S}$ with identifiers $I = \{0, \ldots, n-1\}$, identical agents of the form $A = (\mathbb{R}^2, \mathbb{R}^2, \mathbb{R}^2, f(x,u) = u)$, and r-disk communication edge map E_{cmm}, that is, i and j are connected if $\|x^{[i]} - x^{[j]}\| \le r$. We use several constants to specify values known a priori and used by each agent's control law, message generation function, and state transition function. Pick some lower numbers $d_1 < d_2 \le r$ to be the distance constraints used in practice, and let all robots know d_1 and d_2 before executing the algorithm. Each robot also knows its position $(x_{\text{final}}, y_{\text{final}})$ in the final configuration;

A *formation morphing input-output control and communication law over* $\mathcal{S}$ consists of

(i) communication schedule equal to $\frac{1}{4}nt_{\text{cmm}}$ for $n \in \mathbb{N}$. This implies that t_{cmm} is the time for CONNECTIVITY MAINTENANCE ALGORITHM to go through all four iterations in a cycle. We also choose v_{scale} such that $4v_{\text{scale}}t_{\text{cmm}} < |d_2 - d_1|$;

(ii) communication language $L = \mathbb{R}^2 \times W^{[i]}$;

(iii) logic variables, $W^{[i]} = \mathbb{Z}_n \times \{\text{true}, \text{false}\} \times \mathbb{Z}_n \times \mathbb{N}^3 \times \mathbb{Z}_3$, $w^{[i]} = (i, \mathbb{I}^{[i]}_{\text{branch}}, n^{[i]}_{\text{chld}}, \text{anc}^{[i]}, M^{[i]}_{\text{mode}})$, where i is the agent unique identifier, $\mathbb{I}^{[i]}_{\text{branch}}$ is a flag used to indicate which stage of the algorithm i is executing and $n^{[i]}_{\text{chld}}$ is the number of children of i in the target tree. Each robot, i, will know enough about its ancestors in the constraint tree, T_{targ}, of the target configuration, $[(P_{\text{targ}}, T_{\text{targ}})]$ to answer membership queries of the form "is j an ancestor of i?" and distance queries of the form "given that j is an ancestor of i, how many edges are in the shortest path from i to j in T_{targ}?" We show in Theorem 4.2 that this information can be stored with three integers which we denote by $\text{anc}^{[i]}$. $M^{[i]}_{\text{mode}} \in \mathbb{Z}_3$ is used to indicate to the control function which of three motion modes to use: M_{stay}, M_{origin} and $M_{\text{u,v final}}$, corresponding to a stationary mode, a "move towards origin" mode and a "move towards final configuration" mode, respectively;

(iv) $W^{[i]}_0 = \{(i, \text{false}, n^{[i]}_{\text{chld}}, \text{anc}^{[i]}, M_{\text{stay}})\} \subset W^{[i]}$;

(v) ${W_{\text{in}}}^{[i]} = \mathbb{Z}_n$, where ${w_{\text{in}}}^{[i]} = p^{[i]}_{\text{curr}}$, indicates the ith robots parent in the constraint tree;

(vi) ${W_{\text{in}}}^{[i]}_0$ such that the tree induced by $p^{[i]}_{\text{curr}}$ is a connected spanning tree of the robotic network;

(vii) ${W_{\text{out}}}^{[i]} = \{{w_{\text{out}}}^{[i]} : \mathbb{Z}_n \mapsto \mathbb{Z}_n \cup \{\infty\}\}$, where ${w_{\text{out}}}^{[i]} \in {W_{\text{out}}}^{[i]}$ is a ranking of which nearby robots i would prefer to be connected to;

and with the following functions

(i) the standard message generation function (i.e., $\text{msg}(t, x^{[\text{id}]}, w^{[\text{id}]}, j) = (x^{[\text{id}]}, w^{[\text{id}]})$);

(ii) **function** $\text{stf}_{\text{out}}(t, x^{[\text{id}]}, p^{[\text{id}]}_{\text{curr}}, (\text{id}, \mathbb{I}^{[\text{id}]}_{\text{branch}}, \text{anc}^{[\text{id}]}, M^{[\text{id}]}_{\text{mode}}), \{l_j \mid j \in \mathcal{N}_{\text{id}}\})$

1: Let ${w_{\text{out}}}^{[\text{id}]} : \mathbb{Z}_n \mapsto \mathbb{Z}_n \cup \{\infty\}$ be defined by ${w_{\text{out}}}^{[\text{id}]}(j)$

$$\begin{cases} \infty, & \|x^{[j]} - x^{[\text{id}]}\| > d_1, \\ k, & j = a_{\text{id},k} \wedge (\mathbb{I}^{[i]}_{\text{branch}} = \mathbb{I}^{[j]}_{\text{branch}} \vee j = p^{[\text{id}]}_{\text{curr}}), \\ m_{\text{id}} + 1, & j \notin \{a_{\text{id},k} \mid k \in 1 \ldots m_{\text{id}}\} \wedge j = p^{[p^{[\text{id}]}_{\text{curr}}]}_{\text{curr}}, \\ m_{\text{id}} + 2, & j \notin \{a_{\text{id},k} \mid k \in 1 \ldots m_{\text{id}}\} \wedge j = p^{[\text{id}]}_{\text{curr}}, \\ \infty, & \text{otherwise.} \end{cases}$$

2: return ${w_{\text{out}}}^{[\text{id}]}$

In other words, if no member of $\{a_{\text{id},k} \mid k \in \{1, \ldots, m_{\text{id}}\}\}$ is available, id would prefer to attach to $p^{[p^{[\text{id}]}_{\text{curr}}]}_{\text{curr}}$, and if $p^{[p^{[\text{id}]}_{\text{curr}}]}_{\text{curr}}$ is also not in reach, id will remain attached to $p^{[\text{id}]}_{\text{curr}}$;

(iii) **function** $\text{stf}_{\text{lv}}(t, p^{[\text{id}]}_{\text{curr}}, {w_{\text{in}}}^{[\text{id}]}, w^{[\text{id}]}, \{l_j \mid j \in \mathcal{N}_{\text{id}}\})$ as defined in Table II;

(iv) the control function is $\text{ctl}(x^{[\text{id}]}, w^{[\text{id}]}) = 0$ if $M^{[\text{id}]}_{\text{mode}} = M_{\text{stay}}$, $\text{ctl}(x^{[\text{id}]}, w^{[\text{id}]}) = v_{\text{scale}}\,\text{vers}(-x^{[\text{id}]})$ if $M^{[\text{id}]}_{\text{mode}} = M_{\text{origin}}$, and $\text{ctl}(x^{[\text{id}]}, w^{[\text{id}]}) = v_{\text{scale}}\,\text{vers}(x_{\text{final}} - x^{[\text{id}]})$ if $M^{[\text{id}]}_{\text{mode}} = M_{\text{u,v final}}$.

Next, we specify how the numbers $\text{anc}^{[i]} \in \mathbb{N}^3$ are initialized.

Prior to running any control algorithms, perform the following operations on the constraint tree,

function $\text{stf}_{\text{lv}}(t, p_{\text{curr}}^{[\text{id}]}, w_{\text{in}}{}^{[\text{id}]}, w^{[\text{id}]}, \{l_j \mid j \in \mathcal{N}_{\text{id}}\})$

1: **if** $\|x^{[\text{id}]} - x^{[p_{\text{curr}}^{[\text{id}]}]}\| \geq d_1$ **then**
2: Set $M_{\text{mode}}^{[\text{id}]} \leftarrow M_{\text{parent}}$
3: **else if** $\mathbb{I}_{\text{branch}}^{[\text{id}]} \neq$ true **then**
4: **if** $p_{\text{curr}}^{[\text{id}]} = a_{\text{id},1}$ **then**
5: **if** each j such that $p_{\text{curr}}^{[j]} =$ id has sent $\mathbb{I}_{\text{branch}}^{[j]} =$ true and there are $n_{\text{chld}}^{[\text{id}]}$ such j **then**
6: Set $\mathbb{I}_{\text{branch}}^{[\text{id}]} \leftarrow$ true
7: $M_{\text{mode}}^{[\text{id}]} \leftarrow M_{\text{origin}}$
8: **else if** $\mathbb{I}_{\text{branch}}^{[\text{id}]} =$ true **then**
9: $M_{\text{mode}}^{[\text{id}]} \leftarrow M_{\text{u,v final}}$
10: return $(\text{id}, \mathbb{I}_{\text{branch}}^{[\text{id}]}, n_{\text{chld}}^{[\text{id}]}, \text{anc}^{[\text{id}]}, M_{\text{mode}}^{[\text{id}]})$

TABLE II

stf_{lv} FOR FORMATION MORPHING ALGORITHM.

T_{targ}, of $[(P_{\text{targ}}, T_{\text{targ}})]$. Perform a depth-first search on T_{targ}. Mark each node, i, with the number of nodes $n_{\text{visit}}(i)$ visited before node i and the number of descendants $n_{\text{desc}}(i)$ of i in the tree T_{targ} . Note that $n_{\text{visit}}(i) + n_{\text{desc}}(i)$ is the number of nodes visited before the first node after i that is not an ancestor of i is visited. Recalling that m_i is the depth in the final target tree of node i, let $\text{anc}^{[i]} \leftarrow (n_{\text{visit}}(i), n_{\text{desc}}(i), m_i)$.

Theorem 4.2: The numbers $\text{anc}^{[i]} \in \mathbb{N}^3$, $i \in I$, allow FORMATION MORPHING ALGORITHM to answer queries of the form "Is robot j $a_{i,d}$ in $[(P_{\text{targ}}, T_{\text{targ}})]$?" using only $O(\log(n))$ bits of storage in $O(1)$ time.

Note that storing a unique identifier for each robot requires $O(\log(n))$ bits. The FORMATION MORPHING ALGORITHM is the composition (in the sense of Definition 2.4) of CONNECTIVITY MAINTENANCE ALGORITHM with the formation morphing input-output law defined above.

B. Correctness analysis

We now establish the correctness of FORMATION MORPHING ALGORITHM. Lemma 4.3 shows that we do not break any edges in the constraint tree.

Lemma 4.3: While following FORMATION MORPHING ALGORITHM composed with CONNECTIVITY MAINTENANCE ALGORITHM, no two robots that are connected in the constraint tree are ever d_2 apart.

Proof: Let robot i be the parent of robot j in the constraint tree. Let t_0 be the first instant of time at which this edge is contained in the tree. Consider the distance $d_{ij}(t) = \|x^{[i]}(t) - x^{[j]}(t)\|$. If this edge is the result of a re-attach event, then d_{ij} must have been less than d_1 one round before the attach (by line 15 of CONNECTIVITY MAINTENANCE ALGORITHM, stf_{out} of FORMATION MORPHING ALGORITHM and the fact that the connect step is one round later at line 3 of CONNECTIVITY MAINTENANCE ALGORITHM). By the definition of v_{scale}, j and i cannot have moved more than $\|d_2 - d_1\|$ further apart in the intervening round. If t_0 is the initial creation of the spanning tree, then we still have $\|x^{[i]}(t_0) - x^{[j]}(t_0)\| < d_1$. At every round of communication one of two things happens. If $d_{ij}(t) \leq d_1$, the robots cannot move more than $2v_{\text{scale}}t_{\text{cmm}}$ further apart before the next communication round. Since $v_{\text{scale}} \leq \frac{d_2 - d_1}{2t_{\text{cmm}}}$, d_{ij} will be less than d_2 by the next communication round. On the other hand, if $d_{ij}(t) > d_1$, robot j moves towards $x^{[i]}(t)$ until the next communication round with velocity v_{scale}. Since i is moving with velocity at most v_{scale}, d_{ij} can be at most $2v_{\text{scale}}t_{\text{cmm}} - d_1$ away by the next communication round. Since $v_{\text{scale}} \leq \frac{d_1}{2t_{\text{cmm}}}$, d_{ij} will be smaller than d_2 at next communication round. ■

Lemma 4.4 and Lemma 4.5 establish that FORMATION MORPHING ALGORITHM performs the necessary topology re-arrangements for formation morphing.

Lemma 4.4: In the execution of FORMATION MORPHING ALGORITHM for each robot, i, the following hold:

- For each descendant, j, of i, $\mathbb{I}_{\text{branch}}^{[i]} =$ true at time t_1 implies $\mathbb{I}_{\text{branch}}^{[j]} =$ true for all $t \geq t_1$.
- $\mathbb{I}_{\text{branch}}^{[i]} =$ true implies for each descendant, j, of i in T_{targ}, $p_{\text{curr}}^{[j]} = a_{j,1}$.

Proof: If any descendant, j, does not satisfy $p_{\text{curr}}^{[j]} = a_{j,1}$, then $\mathbb{I}_{\text{branch}}^{[j]} \neq$ true. Since j cannot attach to any node k having $\mathbb{I}_{\text{branch}}^{[k]} \neq \mathbb{I}_{\text{branch}}^{[j]}$, and $\mathbb{I}_{\text{branch}}^{[p_{\text{curr}}^{[j]}]}$ cannot be set to true until $\mathbb{I}_{\text{branch}}^{[j]} =$ true, no robot, k, in the path from j to the root can have $\mathbb{I}_{\text{branch}}^{[k]} =$ true and neither can i.

If $\mathbb{I}_{\text{branch}}^{[i]} =$ true, each robot j which is a descendant of i in the constraint tree satisfies $p_{\text{curr}}^{[j]} = a_{j,1}$. Each such robot must also be a descendant of i in the target tree, otherwise the path from some j to i would contain a link from k to $p_{\text{curr}}^{[k]}$ where $p_{\text{curr}}^{[k]} \neq a_{k,1}$. Since each descendant, j, of i checks that it has $n_{\text{chld}}^{[j]}$ children before setting $\mathbb{I}_{\text{branch}}^{[j]} =$ true, and no j having $p_{\text{curr}}^{[j]} = a_{j,1}$ switches parents, the number of descendants of i in the constraint tree at time t is equal to the number of descendants in the target tree. ■

Lemma 4.5: Let $\text{diam}(P(t_0))$ be the initial diameter of the convex hull of the robot positions.

(i) Within $O(\frac{\text{diam}(P(t_0))}{t_{\text{cmm}} v_{\text{scale}}})$ rounds, each robot i is within d_1 of each robot in the current path from i to $a_{i,1}$, or satisfies $p_{\text{curr}}^{[i]} = a_{i,1}$.
(ii) Within $4K$ further rounds, each robot, i, is at depth and depth estimate of at least $\min\{\text{dp}_{T_{\text{targ}}}^{[i]}, \max\{K - 2\text{dp}_{T_{\text{targ}}}^{[i]}, 1\}\}$.

Proof: Any robot i with $\mathbb{I}_{\text{branch}}^{[i]} \neq$ true moves towards $(0, 0)$. $O(\frac{\text{diam}(P(t_0))}{t_{\text{cmm}} v_{\text{scale}}})$ is the number of rounds required for the robots to rendezvous under this behavior. Clearly the condition $\|x^{[i]} - k\| < d_1$ holds for all i, and k having $\mathbb{I}_{\text{branch}}^{[i]} = \mathbb{I}_{\text{branch}}^{[k]} =$ false before rendezvous occurs. Each robot, k, along the path from i to the root satisfies $\mathbb{I}_{\text{branch}}^{[k]} =$ false as $p_{\text{curr}}^{[i]} \neq a_{i,1}$. Each robot k along the path from the root to $a_{i,1}$ satisfies $\mathbb{I}_{\text{branch}}^{[k]} =$ false for the same reason. This finishes part 1. We prove part 2 by induction on K and $\text{dp}_{T_{\text{targ}}}^{[i]}$. As a base case, when $K = 0$, every non-root robot is at depth at least 1. Assume true for $K-1$, thus after $4(K-1)$ rounds, for each of i's final ancestors, $a_{i,k}$ (where k is the final distance from $a_{i,k}$ to i) was at depth at least $\min(\text{dp}_{T_{\text{targ}}}^{[a_{i,k}]}, \max(K - 1 - \text{dp}_{T_{\text{targ}}}^{[a_{i,k}]}, 1)) = \min(\text{dp}_{T_{\text{targ}}}^{[a_{i,k}]}, \max(K - \text{dp}_{T_{\text{targ}}}^{[i]} + (2k-1), 1))$.

If $\max(K - \text{dp}^{[i]}_{T_{\text{targ}}}, 1) = 1$ or if i is at its final depth, there is nothing left to do, otherwise each of i's ancestors is at a depth greater than i, or at its final position. This means that the one ancestor of i at i's current depth and depth estimate is in its final position, and line 2 of stf of CONNECTIVITY MAINTENANCE ALGORITHM allows i to make the connection, increase its depth by 1 and over the next 4 rounds, increase its depth estimate to the proper value. ■

Lemma 4.6: Within $\text{dp}^{[i]}_{T_{\text{targ}}}(1+\frac{2d_1}{t_{\text{cmm}}v_{\text{scale}}})$ rounds of the first time $\mathbb{I}^{[j]}_{\text{branch}}$ is true for all j, where T_{targ} is the topology of the final configuration, each robot i is at its final position.

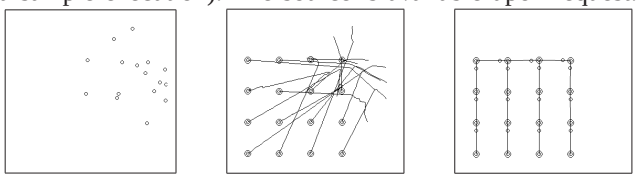
Proof: Let us induct on $\text{dp}^{[i]}_{T_{\text{targ}}}$. As a base case, the root reaches its final position in zero rounds. When $p^{[i]}_{\text{curr}}$ reaches its final position, it stops, and within one more round, i is within d_1 of $x^{[p^{[i]}_{\text{curr}}]}$ which is within a distance of d_1 of i's final position. It takes i at most $\frac{2d_1}{t_{\text{cmm}}v_{\text{scale}}}$ further rounds to reach its final position. ■

These topology lemmas and Lemma 4.6 lead up to Theorem 4.7 which establishes the correctness and time complexity of FORMATION MORPHING ALGORITHM.

Theorem 4.7: Within $O(\frac{(\text{diam}(P(t_0))+\text{diam}(T_{\text{targ}})d_1}{t_{\text{cmm}}v_{\text{scale}}})$ rounds FORMATION MORPHING ALGORITHM achieves formation morphing, where T_{targ} is the final tree in the target configuration, and $\text{diam}(T_{\text{targ}})$ is its graph diameter.

V. SIMULATION RESULTS

We have developed a custom java simulation engine for robotic networks expressed in the formalism of [9]. We used this framework to develop simulations and visualizations of the FORMATION MORPHING ALGORITHM (Figure 1 shows a sample execution). The source is available upon request.



(a) (b) (c)

Fig. 1. Plots show (a) the initial positions, (b) the paths taken by and (c) the final configuration (including constraint tree) of an execution of FORMATION MORPHING ALGORITHM.

We further developed the simulator to run approximately 80000 runs of FORMATION MORPHING ALGORITHM with initial and final configurations sampled randomly. In Figure 2 we plot the actual task completion times of each of these runs versus the function $\text{diam}(P(t_0)) + \text{diam}(T_{\text{targ}})$. Because of the uniform time schedule, the number of communication rounds required for completion is linearly related to the time required for completion. From the graph one can see a linear relationship between the worst completion times for FORMATION MORPHING ALGORITHM and $\text{diam}(P(t_0)) + \text{diam}(T_{\text{targ}})$, as fore-casted by our analysis, see Theorem 4.7.

VI. CONCLUSIONS AND FUTURE WORK

We have introduced the CONNECTIVITY MAINTENANCE ALGORITHM and shown its desirable properties as an extension to existing connectivity maintenance mechanisms for

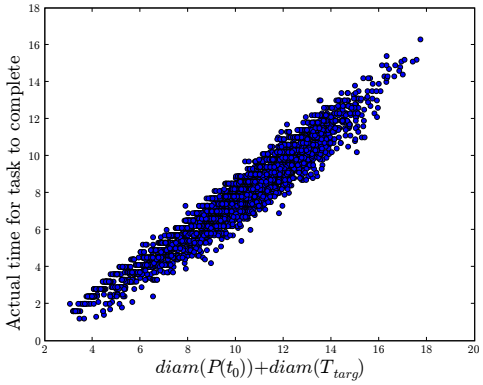


Fig. 2. Comparison of the time complexity bound in Theorem 4.7 with actual running times of FORMATION MORPHING ALGORITHM under random choices of initial and final configurations. Each point represents a successful execution.

distributed control of robotic networks. We have used the CONNECTIVITY MAINTENANCE ALGORITHM to create the FORMATION MORPHING ALGORITHM, which is guaranteed to perform formation morphing while maintaining network connectivity. Finally, we have also characterized the performance of this algorithm by analyzing the time complexity of its execution as the number of agents grow.

Future work will improve the robustness of these algorithms, while loosening the strict synchronization requirements currently in place. We will also explore the interconnection of the CONNECTIVITY MAINTENANCE ALGORITHM with other algorithms to achieve more complex coordination tasks. Natural candidates are multiple-leader leader-follower behaviors, and sensor coverage tasks for mobile sensor networks. We will also investigate the addition of more operations to CONNECTIVITY MAINTENANCE ALGORITHM that still guarantee distributed correctness (e.g., "identity swap" operations between robots). Regarding the FORMATION MORPHING ALGORITHM, we will explore the incorporation of collision-free guarantees on its execution.

ACKNOWLEDGMENTS

This research was supported in part by NSF CAREER Award ECS-0546871.

REFERENCES

[1] S. Boyd, "Convex optimization of graph Laplacian eigenvalues," in *Proceedings of the International Congress of Mathematicians*, vol. 3, 2006.

[2] Y. Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 116–120, 2006.

[3] M. C. D. Gennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *IEEE Conf. on Decision and Control*, San Diego, CA, Dec. 2006, pp. 3628–3633.

[4] M. M. Zavlanos and G. J. Pappas, "Controlling connectivity of dynamic graphs," in *IEEE Conf. on Decision and Control*, Seville, Spain, 2005, pp. 6388–6393.

[5] G. Notarstefano, K. Savla, F. Bullo, and A. Jadbabaie, "Maintaining limited-range connectivity among second-order agents," in *American Control Conference*, Minneapolis, MN, June 2006, pp. 2124–2129.

[6] J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 905–908, 2001.

[7] D. P. Spanos and R. M. Murray, "Motion planning with wireless network constraints," in *American Control Conference*, Portland, OR, June 2005, pp. 87–92.

[8] F. C. Gaertner, "A Survey of Self-Stabilizing Spanning-Tree Construction Algorithms," Tech. Rep., 2003, available electronically at `http://infoscience.epfl.ch/search.py?recid=52545`.

[9] S. Martínez, F. Bullo, J. Cortés, and E. Frazzoli, "Synchronous robotic networks and complexity of control and communication laws," Jan. 2005, preprint. Available electronically at `http://arxiv.org/math.OC/0501499`.

[10] R. Diestel, *Graph Theory*, 2nd ed., ser. Graduate Texts in Mathematics. New York: Springer Verlag, 2000, vol. 173.

[11] C. D. Godsil and G. F. Royle, *Algebraic Graph Theory*, ser. Graduate Texts in Mathematics. New York: Springer Verlag, 2001, vol. 207.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[13] J. W. Jaromczyk and G. T. Toussaint, "Relative neighborhood graphs and their relatives," *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1502–1517, 1992.

[14] J. Cortés, S. Martínez, and F. Bullo, "Robust rendezvous for mobile networks via proximity graphs in arbitrary dimensions," *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1289–1298, 2006.

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