Establishing Connectivity in Proximity Networks

Ran Dai, Joshua Maximoff and Mehran Mesbahi

Abstract—We examine the problem of designing the optimal paths to establish connectivity in a network of initially scattered dynamic agents, specifically minimizing the squared integral of the total control effort. The network edges are modeled by proximity relationships between endpoint agents, leading to a dynamic, state-dependent network topology. We formulate an optimal control problem with specified initial states, linear dynamics, and a connectivity constraint on the final induced topology. Our approach utilizes the Hamiltonian and resultant Euler-Lagrange equations to restructure the optimal control formulation as a parameter optimization problem based on final agent states. We provide both a heuristic approach and an iterative semidefinite programming relaxation to efficiently approximate a solution of the resulting combinatorial optimization problem. Simulation results are provided (for double integrator agent dynamics) to demonstrate feasibility for both approaches, and the results are compared with those obtained from exhaustive global search and random sampling.

Index Terms—Network Topology Design; Proximity Network; Convex Optimization; Cooperative Control; Semidefinite Programming

I. INTRODUCTION

Formation control of connected multi-agent networks has emerged in a variety of areas where cooperative work is required to achieve a common goal. For example, unmanned aerial vehicle (UAV) flocking [1], spacecraft formation [2], and satellite clustering [3] require agents to cooperate to achieve a global objective. A common necessary condition for achieving the specified system goal in these settings is that the underlying communication or sensing graph be connected, if not for all time, then at least in some generalized sense [4].

Motivated by the importance of network connectivity in applications of cooperative control, we consider the problem of generating optimal paths that bring initially scattered agents into proximity in order to establish a connected network. This type of problem has many potential applications for real-world scenarios. For example, a rescue team may need to establish the minimum length paths to connect scattered members in a network where each member has a limited communication radius. Another example is the problem of aligning attitudes in a cluster of satellites with constrained communication angles so that the network becomes connected. The focus in these kinds of problems is not only optimizing the paths from initial states to final states, but also evaluating and/or improving the topology of the final connected network, which also affects the performance index. Considering these factors, the problem of establishing connectivity in support of system objectives can be broken down into two parts—trajectory optimization and network topology design.

Trajectory optimization problems have been investigated for many years. Traditional methods generally fall into two categories—the direct method [5] and the indirect method [6]. Much work has been done in the area of optimal topology design for dynamic networks, with a focus on improving the significant properties of the network topology, e.g., algebraic connectivity [7], resistance [8], and the gap between eigenvalues of Laplacian [9]. For proximity-based networks, approaches that represent the communication strength between agents by a power function that approximates the binary linkage relationship have proved fruitful. In this case, an iterative algorithm can be generated to calculate the ideal position of each agent, leading to the maximization of the algebraic connectivity of the network [10]. However, the cost of moving the nodes to these ideal positions is not considered in these earlier studies. Similar work can be found in [7] and [11] where distributed methods are applied to improve connectivity in an existing connected network.

Our problem formulation differs from much of the previous work in the areas of trajectory and topology design. In our setup, node mobility is the sole means by which network edges can be established, in contrast to variable-communication-radius formulations [12] where the objective is to minimize total transmit power. We specify the minimization of the control effort as the objective, with network connectivity applied as a boundary constraint, in contrast to previous studies that consider optimal ways to maintain or improve the connectivity of a mobile network that is assumed connected at initial time [13].

To solve the optimal path design problem for the formation of a connected network, we employ both the synthesis of trajectory optimization and network topology design techniques. Unfortunately, these two approaches turn out to be highly coupled, which complicates the solution procedure. However, by formulating the path design problem as a parameter optimization problem and converting the connectivity constraint to a semidefinite form, a convex optimization problem can be formulated, which can be solved by a semidefinite programming (SDP) solver in a compact iterative scheme. This iterative approach starts with a fully connected graph and gradually improves the performance index until convergence is achieved. Our alternative approach, a local search heuristic, relies on iteratively choosing edges according to a relative weighting scheme that considers the control effort required to bring the endpoint agents into proximity. We redefine the weighting scheme at each iteration to take into account the additional effort of collectively moving a previously established connected component toward another connected component, which may be required in order to avoid destroying established edges.

The organization of the paper is as follows. Some background information and the initial formulation of the problem is presented in §II. The formulation from §II is transformed into a parameter optimization problem in §III. We present a heuristic search scheme in §IV and an iterative SDP algorithm in §V, both of which exploit the parameter optimization form. Simulation examples demonstrating the feasibility and accuracy of the proposed approaches are detailed in §VI. We conclude the paper with a few remarks in §VII.

II. BACKGROUND AND PROBLEM FORMULATION

We consider a group of $n$ agents initially scattered in space as illustrated in Figure 1. Each agent $i, i \in \{1, 2, \ldots, n\}$, has linear dynamics of the form $\dot{x}_i = A_i x_i + B_i u_i$, where $x_i \in \mathbb{R}^m$ represents the agent’s physical states, e.g., position, angle, velocity, etc, $u_i \in \mathbb{R}^p$ is the control, $A_i \in \mathbb{R}^{m \times m}$ is the linear state
transient matrix, and \( B_\ell \in \mathbb{R}^{m \times p} \) is the linear control input matrix. We assume throughout the paper that the agents are individually controllable as defined in [14], so that \((A_i, B_i)\) forms a controllable pair for each \(i\). The initial states of each agent are given as \( x_{0i} \). The objective is to determine paths that bring these agents into proximity to form a connected network at terminal time \( t_f \), minimizing the total control effort necessary to do so. In Figure 1, the lines with double arrows indicate an edge between two agents at time \( t_f \). This example illustrates just one of the exponential number of possible connected topologies of the final network.

Before specifying the problem, we introduce some notation. We denote the connected network at final time \( t_f \) by simple, undirected graph \( \mathcal{G} = (V, E) \), where the vertex set \( V \) represents the \( n \) mobile agents, and edge set \( E \) is composed by two element subsets of \( V \). The edges of \( \mathcal{G} \) determine an \( n \times n \) adjacency matrix \( A(\mathcal{G}) \) with entries \( a_{ij} = 1 \) when \((v_i, v_j) \in E \) and \( a_{ij} = 0 \) otherwise. Since the graph is undirected, \( A \) is symmetric and because it is simple \( a_{ii} = 0 \) for all \( i \in \{1, \ldots, n\} \). The degree matrix \( \Delta(\mathcal{G}) \) of the graph is a diagonal matrix given by \( \Delta(\mathcal{G})_{ii} = \sum_{j=1}^{n} a_{ij} \) and \( \Delta(\mathcal{G})_{ij} = 0 \) \((i \neq j)\). The adjacency and degree matrices can be combined to form the Laplacian matrix \( \mathcal{L}(\mathcal{G}) = \Delta(\mathcal{G}) - A(\mathcal{G}) \). The eigenvalues of \( \mathcal{L}(\mathcal{G}) \) play an important role in many network problems, including ours. We denote the eigenvalues of \( \mathcal{L}(\mathcal{G}) \) by \( 0 = \lambda_1(\mathcal{G}) \leq \lambda_2(\mathcal{G}) \leq \cdots \leq \lambda_n(\mathcal{G}) \) and therefore have \( \mathcal{L}(\mathcal{G}) \succeq 0 \), where \( \succeq \) denotes the positive semidefinite ordering. It is well known that \( \lambda_2(\mathcal{G}) > 0 \) is a necessary and sufficient condition for connectivity of \( \mathcal{G} \) [15].

Our network model assumes that the connection between two adjacent nodes is determined by their Euclidean distance. Mathematically, we express the binary edge rule by

\[
    a_{ij}(\mathcal{G}) = \begin{cases} 
    1, & \text{if } d_{ij} \leq \epsilon, \\
    0, & \text{if } d_{ij} > \epsilon, 
    \end{cases} \quad \forall i,j \in \{1, \ldots, n\}, \tag{2.1} 
\]

where \( d_{ij} \) is the Euclidean distance between agent \(i\) and \(j\) determined by the states of these agents at time \( t_f \) and \( \epsilon \) is a fixed connection range threshold. This kind of state-dependent network model is often refereed to as a proximity network in mobile ad-hoc networking research.

With proper notation in place, the central problem of the paper can be expressed as

\[
    \min_{u_{1}, \ldots, u_{n}} \quad \sum_{i=1}^{n} \int_{0}^{t_f} \frac{1}{2} a_{ii} u_{ii} dt, \tag{2.2} 
\]

\[
\text{s.t.} \quad x_i = A_i x_i + B_i u_i, \quad i \in \{1, \ldots, n\}, \tag{2.3} 
\]

\[
    x_i(t_0) = x_{0i}, \tag{2.4} 
\]

\[
    a_{ij}(\mathcal{G}) = \begin{cases} 
    1, & \text{if } d_{ij}(x_i(t_f), x_j(t_f)) \leq \epsilon, \\
    0, & \text{if } d_{ij}(x_i(t_f), x_j(t_f)) > \epsilon, 
    \end{cases} \tag{2.5} 
\]

\[
\lambda_2(\mathcal{G}) > 0. \tag{2.6} 
\]

III. PARAMETER OPTIMIZATION PROBLEM

The problem described in (2.2)-(2.6) specifies optimization of a quadratic objective function with linear and nonlinear constraints. The eigenvalue constraint on the second smallest eigenvalue of the Laplacian implicitly enforces nonlinear constraints on the terminal agent states and complicates the solution. We proceed by dividing the approach into two steps. First, the relationship between the optimal controls and the final states of an agent is determined. The Hamiltonian for the optimization problem with objective (2.2), dynamics (2.4), initial constraints (2.5) and terminal state constraint \( x(t_f) \) is formulated as

\[
    H = \frac{1}{2} u^T u + \lambda^T(A x + B u) \tag{3.7} 
\]

where \( \lambda = \lambda(t) \) is the \(m\)-dimensional Lagrangian multiplier vector. Following well-established techniques from dynamic optimization [16], the Euler-Lagrange equations are calculated as

\[
    \dot{x}^T = - \frac{\partial H}{\partial x} = - \lambda^T A \tag{3.8} 
\]

\[
    \frac{\partial H}{\partial u} = u + B^T \lambda = 0. \tag{3.9} 
\]

By solving the above equations, we get

\[
    u(t) = - B^T e^{-A^T t} \lambda_0, \tag{3.10} 
\]

where \( \lambda_0 \) is the Lagrange multiplier defined at the initial time \( t_0 \). Substituting (3.10) into the system dynamics and integrating the states from \( t_0 \) to \( t_f \), the final state \( x_f \) at \( t_f \) can be expressed as

\[
    x_f = e^{A(t_f-t_0)} x_0 - W_c(t_0, t_f) e^{-A^T t_f} \lambda_0, \tag{3.11} 
\]

where the controllability Gramian \( W_c(t_0, t_f) = \int_{t_0}^{t_f} e^{A(t_f-t)} B B^T e^{A^T(t-t_0)} dt \) is invertible by our assumption that the agents are controllable [14]. Since \( W_c(t_0, t_f) \) is nonsingular for a controllable system (2.4), \( \lambda_0 \) can be solved as

\[
    \lambda_0 = f(x_0, x_f) \quad \text{for given } x_f \text{ and } x_0 \quad \text{by} \tag{3.12}
\]

Finally, the optimal controls in (3.10) can be rewritten as

\[
    u = - B^T e^{-A^T t_f} (W_c(t_0, t_f) e^{-A^T t_f})^{-1} (e^{A(t_f-t_0)} x_0 - x_f). \tag{3.13} 
\]

For fixed initial condition and dynamics, the only variable in (3.13) is \( x_f \).

From the above discussion, we find the relationship between the optimal controls and the boundary conditions of the states. When initial condition \( x_0 \) is given, the optimal control trajectory is determined solely by \( x_f \). Proceeding from this observation, the original objective function (2.2) can be simplified as

\[
    J = \sum_{i=1}^{n} \frac{1}{2} \left( e^{A_i t_f} x_{0i} - x_{f_i} \right)^T P_i \left( e^{A_i t_f} x_{0i} - x_{f_i} \right), \tag{3.14} 
\]

where \( P_i = \int_{t_0}^{t_f} Q_i^T Q_i dt \) and \( Q_i \) is defined by

\[
    Q_i = B_i^T e^{-A_i^T t_f} (W_c(t_f) e^{-A_i^T t_f})^{-1}. \tag{3.15} 
\]

We have reduced the search for the optimal (infinite dimensional) continuous-time control trajectory to a simplified finite dimensional search for the optimal final states \( x_{f_i} \). Once the final states are determined, the entire control trajectory can be obtained from (3.10) and (3.11).
IV. Heuristic Approach

In this section we present an iterative graph search heuristic for choosing \( \{x_i^k : i = 1, \ldots, n\} \) based on an equivalent reformulation of (2.2)-(2.6) that leverages the parameter optimization form of the objective from (3.14). The optimal path problem is shown to be reducible to a quadratic program constrained by a union of convex sets, i.e., a generalized disjunctive program (or equivalently as a mixed-integer nonlinear program (MINLP) [17]). Unfortunately, the number of constraints in the reformulation is exponential in \( n \), so standard global MINLP techniques do not scale efficiently as the number of agents increases. Commonly, these kinds of combinatorial optimization problems must be solved approximately either by a local search heuristic (as presented in this section) or some form of relaxation (as presented in the subsequent section).

To motivate the heuristic approach, consider again the equivalent formulation of the original objective function as given in (3.14). By introducing the stacked system state vector \( x = [x_1, \ldots, x_n]^T \), block diagonal system dynamics matrices \( A = \text{diag}(A_1, \ldots, A_n) \) and \( B = \text{diag}(B_1, \ldots, B_n) \), and block diagonal objective matrix \( P = \text{diag}(P_1, \ldots, P_n) \), we can state the original optimization problem as

\[
\min_{x_f \in \mathbb{R}^{m \times n}} \ |D| \left( e^{At_f}x_0 - x_f \right) |_2 \quad (4.16)
\]

\[
s.t. \quad \lambda_2(\mathcal{G}) > 0, \quad (4.17)
\]

where \( D = \text{diag}(D_1, \ldots, D_n) \) and each \( D_i = P_i^{1/2} \) is determined by the Cholesky factorization of \( P_i \).

Cayley’s theorem specifies that the number of distinct labeled trees on \( n \) nodes is \( n^{n-2} \). Since each connected graph spans at least one tree, and each tree is fully specified by \( n-1 \) convex proximity constraints of the form \( d_{ij} \leq \epsilon \), constraint (4.17) is equivalent to the disjunctive constraint given by

\[
\bigvee_{T \in \mathcal{T}} \bigwedge_{e_{ij} \in E(T)} d_{ij} \leq \epsilon, \quad (4.18)
\]

where \( \mathcal{T} \) denotes the set of all labeled trees on \( n \) nodes and \( E(T) \) is the edge set of tree \( T \).

In principle, it is possible to find a global optimum \( x_f^* \) and a corresponding optimal control law from (3.10) and (3.12) by solving exhaustively over each element of \( \mathcal{T} \), and choosing the minimum among all solutions. Of course, the exponential size of \( \mathcal{T} \) means that this is impractical except for very small \( n \). Our heuristic approach leads to the selection of a \( \{e_{ij}\} \) (but not necessarily optimal) target tree \( T \), and proceeds to minimize the convex, finite objective (4.16) subject to the \( n-1 \) convex constraints defined by the edges of \( T \), all of which can be done in polynomial time.

From (4.16) and (4.17) it is evident that an optimal final state \( x_f^* \) is one for which \( |D(e^{At_f}x_0 - x_f^*)|_2 \) is as small as possible while guaranteeing that \( \mathcal{G} \) is connected. This means that each eigenvalue of \( D \) can be viewed as the relative control cost of changing the terminal system state from the zero-input terminal state by the corresponding unit eigenvector (with smaller eigenvalues indicating larger control costs). Since \( D \) is a block diagonal matrix composed of \( n \) upper triangular blocks (each of dimension \( m \times m \)), the eigenpairs of \( D \) are naturally associated with individual agents. From a given system state \( x \), we form a relative weighting scheme on potential edges of a final graph, where the weight \( e_{ij} \) assigned to potential edge \( e_{ij} \) is given by

\[
e_{ij}(x) = \left\| D_{ij} e^{At_f} x_i - D_{ji} e^{At_f} x_j \right\|_2, \quad (4.19)
\]

We refer to this specialized distance between nodes as the control effort separation. It indicates which pairs of nodes are relatively inexpensive to bring together, as measured by their terminal distance under no control, providing a reasonable estimate of the edges induced by an optimal final state.

Using the control effort separation between zero-input terminal node states for edge weights in a complete weighted graph, it is natural to consider a target tree \( T \) by selecting the edges in a minimum-weight spanning tree over the complete graph. However, the control effort separation does not account for the fact that each selected edge requires thereafter that both endpoints move toward any other node incident to either of them in the chosen tree. In other words, \( e_{ij} \) ignores the cost of maintaining any edge \( e_{ik}, \ k \neq j \) as node \( i \) and node \( j \) are brought into proximity. This suggests a refined strategy based on recomputing the weights on potential edges iteratively while adding edges one at a time. The general idea is to select at each iteration the edge with least control effort separation that also improves connectivity of the graph. Then the agent states are propagated forward by solving (4.16) subject to the current set of edge constraints to establish the optimal relative positions that induce the specified edges. Recomputing the control effort separation for these virtual agent states and selecting the next edge as a minimization of the new weightings adds memory to the algorithm in the sense that the weights in a given iteration take into account the states of the nodes after inducing previous edges. Algorithm iterative-tree-build (ITB) below makes this strategy explicit.

**Algorithm iterative-tree-build (ITB)**

**Input:** System dynamics parameters \( A, B, x_0, t_f \)

**Output:** Terminal state \( x_f \) and target tree \( T \)

begin

1) initialize \( x^{(1)} = e^{At_f} x_0, E^{(1)} = E(G(x^{(1)})) \)

2) while \( G(x^{(k)}) \) is not connected

3) \( E^{(k+1)} = E^{(k)} \cup \{e_{ij}\} \), where \( e_{ij} \) has the following properties:

   a) \( e_{ij} \notin E^{(k)} \)

   b) Adding \( e_{ij} \) reduces the number of components of \( G(x^{(k)}) \)

   c) Subject to (a) and (b), \( e_{ij}(x^{(k)}) \) is as small as possible

4) Minimize (4.16) over the constraint set indicated by \( E^{(k+1)} \)

   a) Update state \( x^{(k+1)} \) as the resultant optimal terminal state

   b) \( k = k + 1 \)

5) end while

6) Set \( x_f = x^{(k)}, T = (V, E^{(k)}) \)

end

Note that the output graph \( T \) from ITB is indeed a tree. Step 3(b) guarantees no edges that complete a cycle will be considered for addition to \( T \). At any iteration \( k \), \( x^{(k)} \) may induce a superset of the edge set \( E^{(k+1)} \), as we do not require that any internode distances be larger than the edge distance threshold. However, we specifically do not add these edges to \( E^{(k+1)} \) since they are not strictly necessary for the purposes of reducing the component count. In practice, these extra edges tend to have endpoints with small control effort separation in the next iteration of the algorithm, so if any such edge does reduce the number of components, it will likely be a prime candidate for addition in step 4 of a subsequent iteration.
Because each iteration is guaranteed to reduce the component count of the resultant graph, at most \( n - 1 \) iterations of ITB are required to establish graph \( T \). Step 3 requires computing \( c_{ij} \) for all \( \binom{n}{2} \) pairs of agents, \( i \) and \( j \), and finding the smallest value such that the associated edge reduces the component count (which can be computed efficiently from the eigenvalues of the edge-augmented adjacency matrices). The complexity of the entire algorithm, then, is at most a polynomial factor worse than the complexity of minimizing (4.16) subject to \( n - 1 \) intermode distance constraints, which, itself, is a quadratically constrained quadratic program (QCQP) solvable in polynomial time. In other words, ITB can be executed in polynomial time.

V. ITERATIVE SDP ALGORITHM

In this section, we present an iterative SDP based algorithm to approach the optimal solution with fast convergence. The simplified optimization problem focuses on searching the final states to minimize expression (3.14) while satisfying the terminal graph connectivity constraint. The existence of an edge between two agents is an integer variable constrained by their Euclidean distance at time \( t_f \). The elements of the Laplacian for the graph at final time are therefore linear functions of these nonlinear integer variables. The complications introduced by constraining \( \lambda_2 \) for this integer-valued Laplacian matrix classifies the problem as NP-hard, and suggests a relaxed formulation.

The first step in the proposed iterative SDP algorithm is to introduce a special function to approximately express the integer variables constrained by the Euclidean norm. In some situations, the information exchange between agents uses communication devices whose efficacy drops off continuously as the distance between the agents increases (e.g., radar). In these scenarios, it may actually introduce a special function to approximately express the integer valued Laplacian matrix classifies the problem as NP-hard, and suggests a relaxed formulation.

We introduce Finsler’s Lemma to convert the eigenvalue constraint to an equivalent semidefinite constraint.

**Lemma 5.1:** Let \( x \in \mathbb{R}^n \), \( Q \in \mathbb{R}^{n \times n} \) and \( U \in \mathbb{R}^{m \times n} \) such that rank \((U) < n \). The following statements are equivalent:

1) \( x^T Q x > 0 \), for all \( U x = 0, x \neq 0 \).
2) \( U^T Q U > 0 \).
3) \( \exists \mu \in \mathbb{R} \), such that \( Q + \mu U^T U > 0 \).

**Remark 5.2:** \( U^T U \) is a basis for the null space of \( U \). That is, all \( x \neq 0 \) such that \( U x = 0 \) is generated by some \( x \neq 0 \) in the form \( x = U \tilde{z} \).

**Proposition 5.3:** For a graph Laplacian \( L(G) \), we have \( x^T L(G) x > 0 \) for all nonzero \( x \in \mathbb{R}^n \) iff \( \lambda_2 > 0 \).

**Proof:** By Courant-Fischer Formula [18], we have \( \lambda_2 = \min_{\|z\|=1,z \perp x} z^T L x \).

**Corollary 5.4:** For a graph Laplacian \( L(G) \), there \( \exists \mu \) such that \( L(G) + \mu I \) is equivalent to \( x^T L(G) x > 0 \) for all nonzero \( x \in B^2 \). From Proposition (5.3), we find the equivalence between \( \lambda_2 > 0 \) and \( x^T L(G) x > 0 \). Therefore, the equivalence exists between \( \lambda_2 > 0 \) and \( \exists \mu, L(G) + \mu I \) is equivalent to \( x^T L(G) x > 0 \).

From these observations and using continuous edge function (5.20), we can transform the original problem (2.2)-(2.6) into the following parameter optimization problem with a semidefinite constraint.

\[
\min_{x_{f_1}, \ldots, x_{f_n}} \sum_{i=1}^{n} \frac{1}{2} \left( e^{A_1 x_{f_i}} - x_{f_i} \right)^T P \left( e^{A_1 x_{f_i}} - x_{f_i} \right) \quad (5.21)
\]

\[
w_{d_{ij}} = 1/(1 + e^{\alpha (d_{ij} - \rho)}) \quad (5.22)
\]

\[
a_{ij} = w_{d_{ij}}, \forall i, j \in 1, 2, \ldots, n \quad (5.23)
\]

\[
L(G) + \mu I \succeq \xi \quad (5.24)
\]

where \( \xi \in \mathbb{R} \) is a positive small number to guarantee that the weighted graph is connected.

Inspired by the work in [10], the iterative SDP algorithm is initialized with some connected terminal graph and a set of node positions that induce the desired graph. Then a movement is prescribed for each node, which reduces the total control effort while assuring that the resultant graph is also connected. As illustrated in Figure 3(a), where node movement is represented by dashed lines, the agents form a new graph with different topology from the previous iteration. The performance is improved in the sense that the nodes require less total control effort to induce the new topology than the previous topology. In general, the graph topology need not differ between iterations, though the new node states are guaranteed to result in an improved objective value.

![Fig. 3](image-url)

**Fig. 3.** Illustration of iterative SDP method.

Consider a single iteration and denote the squared Euclidean distance for the previous graph as \( d_0 \) and the new one as \( d \).
as shown in Figure 3(a) for some edge in the current topology. A change in Euclidean distance between iterations causes the communication strength \( w(d) \) to change in a complicated, nonlinear way. However, by limiting the movement of each agent sufficiently between iterations, the new \( w(d) \) can be approximately updated by a linearized expression of (5.20) as

\[
w(r) = w(d_o) - w(d_o)^2 \alpha e^{-d_o}(r - d_o) + o(r - d_o). \tag{5.25}
\]

Figure 3(b) shows the linearization procedure of the communication strength function when obtaining the approximate expression of \( w(r) \). The quadratic constraints between Euclidean distances \( r \) and final states \( x \) are given by

\[
\| x_f - x_f \|^2 \leq \tau_{ij}, \ i, j \in 1, 2, \ldots, n. \tag{5.26}
\]

Since \( w(d_{ij}) \) is an element of an adjacency matrix \( A \), we must also constrain \( w(d_{ij}) \) by \( 0 \leq w(d_{ij}) \leq 1 \).

The result of the above analysis is an iterative SDP algorithm where at each step \( k \), the updated states \( x_f \) and the Euclidean distances \( r(k+1) \) at final time are determined by solving

\[
\min_{x_f(k+1), r(k+1)} \sum_{i=1}^{n} \frac{1}{2} \left( e^{A_{ij} x_{0i} - x_f(k+1)} P_i \right) \quad \text{s.t.} \quad \| x_f(k+1) - x_f(k+1) \|^2 \leq \tau(k+1) \tag{5.27}
\]

\[
w(r(k+1)) = w(d_{ij}^{(k)}) - w(d_{ij}^{(k)}) \alpha e^{d_{ij}^{(k)} - \rho (r(k+1) - d_{ij}^{(k)}))} \leq 1 \tag{5.29}
\]

\[
0 \leq w(r_{ij}^{(k+1)}) \leq 1 \tag{5.30}
\]

\[
L(G(k+1)) + \| x_f(k+1) \| \leq \xi. \tag{5.31}
\]

Next, we prove the convexity of the problem formulated above.

**Proposition 5.5:** The problem described in (5.27)-(5.31) is a convex optimization problem.

**Proof:** Each summand in the objective function in (5.27) is a quadratic function, where each \( P_i = \int_0^f Q_i^T Q_i dt \) is the integral of the product of a matrix trajectory with its own transpose, so each \( P_i \) is positive semidefinite. So, the Hessian of each summand of the objective function \( J \) is \( \nabla^2 J(x_f(k+1)) = P \geq 0 \), hence the objective function is convex \( \forall x_f(k+1) \in \mathbb{R} \). The constraints in (5.29) and (5.30) are hyperplane and halfspaces convex constraints, respectively. The constraints in (5.28) and (5.31) are norm cone and positive semidefinite cone constraints, respectively, both of which are convex. This means that the problem described in (5.27)-(5.31) is the minimization of a convex objective function over a set of convex constraints. We can conclude that (5.27)-(5.31) describes a convex optimization problem.

In the iterative approach, an initial value of \( d_{ij}^{(1)} \) is given at the first iteration \( k = 1 \). Generally, a complete connected graph with \( d_{ij}^{(1)} = \epsilon, \forall i, j \in 1, 2, \ldots, n \), is selected as the initial guess. At each iteration, \( x_f(k+1) \) and \( r(k+1) \) are determined and then \( d_{ij}^{(k+1)} \) is updated as \( \| x_f^{(k+1)} - x_f^{(k+1)} \|^2 = d_{ij}^{(k+1)} \). The process is repeated until \( |J^{(k)} - J^{(k+1)}| \leq \delta \), where \( \delta \) is any specified stopping criteria.

### VI. SIMULATION EXAMPLES

Simulations with both five and ten randomly located agents are presented in this section. The agent states evolve according to two-dimensional linear double integrator dynamics given by

\[
A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{6.32}
\]

For each agent, the state \( x \) is represented by the coordinates, \([x_u, p_y]\), and velocity, \([v_u, v_y]\). The control \( u \) is represented by \([u_x, u_y]^T\).

The initial states are randomly chosen for 50 independent trials, and we apply the additional constraint that the agents have zero velocity at the final point in order to keep the formation thereafter. The distance threshold \( \epsilon \) is set as 1 and the final time \( t_f \) is 5.

In the \( n = 5 \) case, exhaustive search over each of the 125 labeled trees is possible, so the true global minimum, denoted by \( J_{opt} \), and the expected (i.e., average) objective values can be obtained for comparison with the objective value achieved by the heuristic. In this case, our heuristic algorithm ITB determines a system trajectory that coincides with the global optimal trajectory in 35 of the 50 trials and outperforms the expected value in all trials, as shown in Figure 4. Due to the the iterative SDP’s interagent distance linearization and semidefinite formulation of the connectivity constraint, it is difficult to directly compare the results with those from exhaustive search in an equivalent way to the ITB-exhaustive comparison. However, results from the iterative SDP appear to be near the global optimal results with small-scale fluctuations. An approximate comparison is shown in Figure 4 together with ITB and global optimal results.

The negative values for the comparison ratio are evidence of the difficulties mentioned, though it is clear that the iterative SDP performs quite well. These iterative SDP simulation results use distance thresholds \( \rho_1 = 1 \) and \( \rho_2 = 1.05 \), respectively, a connectivity threshold \( \xi = 0.1 \), and an iterative stopping criteria \( \delta = 0.01 \). At initialization \( d_{ij}^{(1)} \) is set to 1 for all \( i \neq j \).

![Fig. 4. 5-Node Monte Carlo results.](image-url)
heterogeneous) dynamics. In our simulations of heterogeneous systems, the heuristic performs better than the expected value in a majority trials, though exceptions do occur.

In Figures 6 and 7 the individual agent trajectories as they form the final connected graph are shown for both the iterative SDP algorithm and ITB (for a single trial from the 10-nodes simulation set). The initial position of each agent is labeled by $A_i(0)$, and the trajectories are displayed as circles with edges of the terminal graph displayed as solid lines. Notice that in both cases the agents finally form a tree, i.e., a minimally connected graph, though the topologies are quite different for the two algorithms. In addition, we illustrate the convergence of the solution for a single 10-node case in Figure 8, where $J^*$ denotes the objective value at the iteration for which the algorithm meets the stopping criteria.

VII. CONCLUSIONS

The problem of designing the optimal paths to establish connectivity for a group of initially scattered dynamic agents with limited-range communications approximated via two novel approaches, the iterative-tree-build and the iterative SDP. We used simulation results to validate the efficiency and feasibility of the proposed methods. Differences in the way edges are specified in each of the two approaches make it difficult to compare the results head-to-head to choose a preferred method. Instead, we suggest a careful evaluation of the trade-off between approximation of optimality (offered by the iterative-tree-build algorithm) and constraint relaxation to a solvable convex form (offered by the iterative SDP). The proposed algorithms are of importance in many areas of practical interests, such as gathering scattered rescue team members and satellite formation.

REFERENCES