Behavior Mixing in Multi-Robot Systems

Wenhao Luo, Sha Yi, and Katia Sycara

***Abstract*— Multi-robot systems that employ behavior based control to collaboratively complete complex tasks through local interaction are known as *robot swarms*. In many cases the robots are desired to execute simultaneously multiple behaviors with different controllers, or sequences of behaviors, in real time, which we all *behavior mixing*. Behavior mixing is accomplished when different subgroups/subswarms of the overall swarm change their controllers to accomplish given tasks while maintaining connectivity with the rest of the swarm. While previous research has focused on algorithms for single swarm behavior, or on sequential composition of behaviors, this paper presents *for the first time* provably optimal algorithms for behavior mixing. The technical idea underlying behavior mixing is the ability of the swarm to *stay connected in a flexible way*. Existing algorithms either over restrain connectivity, or introduce large perturbation to the original control outputs. To achieve the needed connectivity flexibility for behavior mixing, we propose a Distributed Minimum Connectivity Constraint Spanning Tree that selects the minimum connectivity constraints in a formally proven *optimal fashion*. We demonstrate the effectiveness and scalability of our approach via simulations of up to 50 robots in presence of multiple behaviors.**

I. Introduction

Distributed multi-robot systems where each robot interacts locally with its neighbors to perform collaborative tasks are known as robotic swarms. Swarms cooperatively exhibit behaviors, such as flocking, rendezvous, formation control etc.[1][2]. The decentralized and scalable nature of robot swarms have large advantages in performing complex tasks including search and rescue, exploration and environment monitoring [2][3][4].

Most work has focused on control laws and optimal performance of swarms that can perform only a single behavior at a time [5][6]. To achieve complex tasks in environments with obstacles, one solution is composing a sequence of single behaviours from a swarm behaviour library [2]. However, there are many situations where it may be more appropriate and efficient to have the swarm in one connected component *simultaneously* performing different behaviors. One example is having a swarm form multiple different formations or rendezvous to multiple places at the same time to track different targets or monitor different areas. To achieve these behaviors simultaneously, different subgroups of the swarm must be involved, each performing a different behavior, or sequences of behaviors as required by the overall task. For example, in order to monitor two areas while the main swarm moves towards a third area, the two subswarms may need to perform a sequence of flocking and rendezvous or flocking and forming a circle around the area.

We call this ability of a swarm to accommodate different behaviors within a single connected swarm while maintaining safety (collision avoidance with other robots and obstacles) and connectivity *Behavior Mixing*. To the best of our knowledge there is no existing work that has the behavior mixing capability. Our framework enables both executing behaviors *in sequence*, and also *in parallel*, so that a single swarm can flexibly subdivide itself depending on the dynamically arising simultaneous tasks. Behavior mixing is very advantageous since (a) it allows the swarm to perform more complex missions, and (b) do so efficiently since the swarm can operate in parallel.

Developing a behavior mixing framework faces many challenges: (a) possible communication disconnection due to performing multiple conflicting behaviors at the same time; (b) the additional constraints imposed on robots with different behaviours by the connectivity controller may lead to behavior failure, for example dead locks that might prevent the desired execution of behaviors; and (c) inefficiency incurred by the perturbation of connectivity on control outputs between different behaviour groups.

To maintain flexible connectivity and avoiding conflict during the execution of different behaviors, we employ control barrier functions [7], [8] that impose safety and connectivity constraints on swarm task-related controllers and ensure the forward invariance of satisfying set. However, the existing control barrier functions on connectivity requires either predefined fixed connectivity topology [9] or enumerating all possible combination of connectivity topology [8]. Such rigid constraints is not scalable nor feasible in behavior mixing as number of robots and behaviors arises. To that end, we propose the idea of *Minimum Connectivity Constraint Spanning Tree (MCCST)* to invoke real-time connectivity constraints that provides provable guarantee on minimizing perturbations from the constraints. The MCCST is constructed from an induced graph of the original connectivity graph, with weights directly related to the strength of connectivity constraints over robot controllers. In this case, by constructing a distributed MCCST to flexibly invoke minimum connectivity constraints across robots, we are able to execute different behaviors simultaneously on a single connected swarm, with optimal connectivity and efficiency in execution and computation. We will first give the formal definition of behavior mixing, and introduce our algorithm of constructing the Minimum Connectivity Constraint Spanning Tree (MCCST) in a *distributed fashion.* Further, we will present our results in evaluating metrics of the MCCST

This work was funded by the DARPA Cooperative Agreement No.: HR00111820051.

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Email: {wenhao, shayi, katia}@cs.cmu.edu.

Preprint submitted to 2019 International Conference on Robotics and Automation (ICRA). Received September 16, 2018.

construction as well as performance of up to 50 robots in simulation.

Our paper presents the following contributions: (1) for the *first* time it presents a framework to simultaneously execute different behaviors, or sequences of behaviors, within a single swarm while ensuring necessary connectivity, while avoiding collisions within the swarm; (2) To enable flexible behavior mixing, we proposed a distributed MCCST approach to efficiently select connectivity constraints with provably optimality guarantee that is minimally invasive to any task-related controllers; (3) the construction of MCCST is *computationally efficient* thus it can be done in real-time to accommodate dynamic changes in the environment; (4) the algorithm is *scalable* to large number of robots.

II. Related Work

Swarm control focuses on robots' distributed incremental movements governed by predefined control laws towards the goal region such as flocking strategy [6], [10], [11] or formation control [12]. To avoid inter-robot collisions and maintain connectivity, auxiliary controllers that handle these constraints are usually combined into frameworks of swarm control. In particular, the existing connectivity control approaches fall into three main categories, i.e., maintaining local connectivity [11], global algebraic connectivity [13] and control barrier function based connectivity [8]. Although the approaches work well for small number of robots with single behavior, they are not capable of ensuring connectivity in multiple behaviors as they do not provide formal guarantee on the perturbation induced from the connectivity controller. We prop[osed] of Minimum Spanning Tree to quantitatively formalize the connectivity constraints and further yield provable optimal constraints selection in favor of behavior mixing.

There has been significant work over the last century on the construction of distributed Minimum Spanning Tree (MST). A significant work from Gallager et al. [14] describes a distributed algorithm for finding MST, by utilizing the parallel nature of Boruvka's [15] algorithm. With different application scenarios, the construction of MST could be time-optimal [16] or message-optimal [17]. Most recent work [18] proposes a time- and message-optimal distributed approach. It utilize the diameter of graph and maintain the diameter of each sub-MST, constructing sub-MST in parallel and speed up by using a randomized algorithm [19] when merging sub-MST with their minimum outgoing edge.

III. Behavior Mixing

Consider a heterogeneous robotic team $\mathcal{S}$ consisting of N mobile robots in a planar space, with the position and single integrator dynamics of each robot $i \in \{1, \dots, n\}$ denoted by $x_i \in \mathbb{R}^2$ and $\dot{x}_i = u_i \in \mathbb{R}^2$ respectively. Each robot can connect and communicate directly with other robots within its spatial proximity. The communication graph of the robotic team is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $v \in \mathcal{V}$ represents a robot. If the spatial distance between robot $v_i \in \mathcal{V}$ and robot $v_j \in \mathcal{V}$ is less or equal to the communication radius R_c (i.e. $\|x_i - x_j\| \leq R_c$), then we assume the two can communicate and edge $(v_i, v_j) \in \mathcal{E}$ is undirected (i.e. $(v_i, v_j) \in \mathcal{E} \Leftrightarrow (v_j, v_i) \in \mathcal{E}$).

In behavior mixing, we assume the robotic team is tasked with M simultaneous behaviors, partitioning the set of robots into M sub-groups $\mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_M\}$. Each sub-group of robots is defined by a tuple $(\mathcal{S}_m, \mathcal{U}_m)$ with $\mathcal{U}_m$ as the corresponding behavior controller and $|\mathcal{S}_m| = N_m$ for all $m = 1, \dots, M$ and $\sum_{m=1}^{M} N_m = N$. To simplify our discussion and focus on behavior mixing, we assume the sub-group partitions and behavior controllers are given, namely, each robot i has been assigned to a sub-group $\mathcal{S}_m$ with some task-related controller $u_i = \hat{u}_i \in \mathcal{U}_m$. To ensure successful behavior mixing, it is required that 1) the communication/connectivity graph $\mathcal{G}$ is connected, and 2) each induced sub-graph $\mathcal{G}_m = \mathcal{G}[\mathcal{V}_m] = (\mathcal{V}_m, \mathcal{E}_m) \subset \mathcal{G}$ where $\mathcal{V}_m \subset \mathcal{V}$ containing robots within the same sub-group $\mathcal{S}_m$ should be connected, for all $m = 1, \dots, M$. This is common as the robotic team needs to exchange information through $\mathcal{G}$ to coordinate their various tasks and inside each sub-group the robots rely on local interaction to achieve desired collective behavior, such as consensus-based rendezvous, flocking, etc.. In presence of the above connectivity constraints as well as the physical constraints of the robots such as inter-robot collision avoidance and velocity limits, each robot i may have to modify their primary task-related controller $\hat{u}_i$ to accommodate the constraints. To that end, the objective of multi-robot behavior mixing is to 1) coordinately invoke active constraints to follow (particularly the connectivity constraints imposed between pair-wise robots), such that the modification to the primary controller is minimum for the robotic team, and 2) compute the modified controller for behavior mixing. In the remaining of this section, we will discuss the formulation of the mentioned constraints in the form of Barrier Certificates on the controllers and will present the formalized optimization problem.

A. Safety and Connectivity Constraints using Barrier Certificates

During movements of multi-robot systems, the robots should avoid collisions with each other to remain safe. Consider the joint robot states $\mathbf{x} = \{x_1, \dots, x_N\} \in \mathbb{R}^{2N}$ and define the minimum inter-robot safe distance as R_s, for any pair-wise inter-robot collision avoidance constraint between robots i and j. We have the following condition defining the safe set of $\mathbf{x}$.

$$\begin{aligned} h^s_{i,j}(\mathbf{x}) &= \|x_i - x_j\|^2 - R_s^2, \qquad \forall i > j \\ \mathcal{H}^s_{i,j} &= \{\mathbf{x} \in \mathbb{R}^{2N} : h^s_{i,j}(\mathbf{x}) \geq 0\} \end{aligned} \tag{1}$$

The set of $\mathcal{H}^s_{i,j}$ indicates the safety set from which robot i and j will never collide. For the entire robotic team, the safety set can be composed as follows.

$$\mathcal{H}^s = \bigcap_{\{v_i, v_j \in \mathcal{V} : i > j\}} \mathcal{H}^s_{i,j} \tag{2}$$

[7] proposed the safety barrier certificates $\mathcal{B}^s(\mathbf{x})$ that map the constrained safety set (2) of $\mathbf{x}$ to the admissible joint control

space $\mathbf{u} \in \mathbb{R}^{2N}$. The result is summarized as follows.

$$\mathcal{B}^s(\mathbf{x}) = \{\mathbf{u} \in \mathbb{R}^{2N} : \dot{h}^s_{i,j}(\mathbf{x}) + \gamma h^s_{i,j}(\mathbf{x}) \geq 0, \forall i > j\} \quad (3)$$

where γ is a user-defined parameter to confine the available sets. It is proven in [7] that the forward invariance of the safety set $\mathcal{H}^s$ is ensured as long as the joint control input $\mathbf{u}$ stays in set $\mathcal{B}^s(\mathbf{x})$. In other words, the robots will always stay safe if they are initially inter-robot collision free and the control input lies in the set $\mathcal{B}^s(\mathbf{x})$. Note that at any time point t with known current robot states $\mathbf{x}(t)$, the constrained control space in (3) corresponds to a class of linear constraints over pair-wise control inputs u_i and u_j for $\forall i > j$.

Next, we consider the pair-wise connectivity constraints among the robotic team. If the connectivity constraint is enforced between pair-wise robots i and j to ensure inter-robot distance not larger than communication range R_c, we have the following condition.

$$\begin{aligned} h^c_{i,j}(\mathbf{x}) &= R_c^2 - \|x_i - x_j\|^2 \\ \mathcal{H}^c_{i,j} &= \{\mathbf{x} \in \mathbb{R}^{2N} : h^c_{i,j}(\mathbf{x}) \geq 0\} \end{aligned} \quad (4)$$

The set of $\mathcal{H}^c_{i,j}$ indicates the feasible set on $\mathbf{x}$ from which robot i and j will never lose connectivity. Consider any connectivity graph $\mathcal{G}^c = (\mathcal{V}, \mathcal{E}^c) \subset \mathcal{G}$ to enforce, the corresponding constrained set can be composed as follows.

$$\mathcal{H}^c(\mathcal{G}^c) = \bigcap_{\{v_i, v_j \in \mathcal{V} : (v_i, v_j) \in \mathcal{E}^c\}} \mathcal{H}^c_{i,j} \quad (5)$$

Similar to the safety barrier certificates in (3), the connectivity barrier certificates are defined as follows and indicate another class of linear constraints over pair-wise control inputs u_i and u_j for $(v_i, v_j) \in \mathcal{E}^c$ at any time point t.

$$\mathcal{B}^c(\mathbf{x}, \mathcal{G}^c) = \{\mathbf{u} \in \mathbb{R}^{2N} : \dot{h}^c_{i,j}(\mathbf{x}) + \gamma h^c_{i,j}(\mathbf{x}) \geq 0, \forall (v_i, v_j) \in \mathcal{E}^c\} \quad (6)$$

B. Objective Function for Behavior Mixing

Consider the mentioned behavior mixing task of M behaviors and sub-groups where the ith robot v_i in current connected robotic team graph $\mathcal{G}$ is assumed to belong to a sub-group $\mathcal{V}_m \subset \mathcal{V}$. In addition, consider that a task-related primary behavior control input $\hat{u}_i \in \mathcal{U}_m$ has been computed. The robotic team needs to determine whether and how to best modify its primary control input in a minimally invasive manner so as to achieve task-related behaviors while ensuring safety and connectivity of both the obtained global graph $\mathcal{G}^c$ containing all N robots and the induced sub-graphs $\mathcal{G}^c[\mathcal{V}_m]$ for all M sub-groups. With the defined forms of constraints in (3) and (6), we formally define the *behavior mixing* problem as follows.

$$\mathbf{u}^* = \underset{\mathcal{G}^c, \mathbf{u}}{\arg\min} \sum_{i=1}^{N} \|u_i - \hat{u}_i\|^2 \quad (7)$$

$$\text{s.t.} \quad \mathcal{G}^c = (\mathcal{V}, \mathcal{E}^c) \subset \mathcal{G} \quad \text{is connected}$$

$$\mathcal{G}_m = \mathcal{G}^c[\mathcal{V}_m] \quad \text{is connected} \quad \forall m = 1, \ldots, M \quad (8)$$

$$\mathbf{u} \in \mathcal{B}^s(\mathbf{x}) \bigcap \mathcal{B}^c(\mathbf{x}, \mathcal{G}^c), \quad \|u_i\| \leq \alpha_i, \forall i = 1, \ldots, N \quad (9)$$

The above Quadratic Programming (QP) optimization problem is to find the optimal active connectivity graph $\mathcal{G}^c$ from current connected robotic team graph $\mathcal{G}$ and the alternative control inputs $\mathbf{u}^* \in \mathbb{R}^{2N}$ so that connectivity, safety and velocity constraints described in (8),(9) are satisfied while ensuring minimally invasive to the primary controller as shown in Equation (7). Recent works [8], [9] have proposed to solve the problem with either fixed static connectivity topology constraints or enumerate compositions of various connectivity topology beforehand. However, these approaches (a) are not capable to deal with behavior mixing where the connectivity constraints need to be generated *in real-time* in a minimally invasive fashion to allow for parallel behaviors, and (b) are not scalable to large numbers of robots, since they need to enumerate all static connectivity constraints beforehand.

In this paper, we propose to decouple the behavior mixing problem into two sub-problems, namely 1) select provable optimal connectivity graph $\mathcal{G}^c = \mathcal{G}^{c*}$ in a distributed manner that invokes minimally invasive connectivity constraints on robotic team, and then 2) solve the optimization problem (7) with the obtained optimal connectivity graph $\mathcal{G}^{c*}$. Such a solution is *the first of its kind* in that it enables behavior mixing with minimum connectivity constraints.

IV. Behavior Mixing using Distributed Selection of Minimum Connectivity Constraints

A. Minimum Connectivity Constraint Spanning Tree (MCCST)

First we consider the sub-problem of selecting optimal connectivity graph $\mathcal{G}^c \subset \mathcal{G}$ in Eq. (7) that introduces minimum connectivity constraints. As each edge $(v_i, v_j) \in \mathcal{E}^c$ in a candidate graph $\mathcal{G}^c$ enforces one pair-wise linear constraint over primary control inputs $\hat{u}_i$ and $\hat{u}_j$ for robot i and j as shown in Eq. (4), the graph $\mathcal{G}^c$ whose edges define the minimum connectivity constraints must exist among the set of all spanning trees $\mathcal{T}$ of the current connected graph $\mathcal{G}$ that have all the vertices $\mathcal{V}$ covered with $N-1$ edges. This is the minimum possible number of edges for $\mathcal{G}^c$ to stay connected. Hence, the problem boils down to find the optimal spanning tree $\mathcal{G}^{c*} = \mathcal{T}^{c*} \in \mathcal{T}$ of $\mathcal{G}$ whose edges invoke the minimum connectivity constraints in the form of (6) over the robots' controllers. To quantify the strength of connectivity constraint by an edge $(v_i, v_j) \in \mathcal{E}$, we introduce the weight assignment defined as follows.

$$w_{i,j} = \dot{h}^c_{i,j}(\mathbf{x}, \hat{u}_i, \hat{u}_j) + \gamma h^c_{i,j}(\mathbf{x}), \forall (v_i, v_j) \in \mathcal{E} \quad (10)$$

Compared to the connectivity constraint in (6), $w_{i,j}$ indicates the violation of the pair-wise connectivity constraint between the two robots, with the higher value of $w_{i,j}$ the weaker the connectivity constraint. To that end, each candidate spanning tree $\mathcal{T}^c \in \mathcal{G}$ is redefined as a weighted spanning tree $\mathcal{T}^c_w = (\mathcal{V}, \mathcal{E}^T, \mathcal{W}^T)$. Hence the optimal connectivity graph $\mathcal{G}^{c*}$ with constraints in (8) can be obtained as follows.

$$\begin{aligned} \mathcal{G}^{c*} &= \underset{\mathcal{T}^c_w \in \mathcal{T}}{\arg\max} \sum_{(v_i, v_j) \in \mathcal{E}^T} w_{i,j} = \underset{\mathcal{T}^c_w \in \mathcal{T}}{\arg\min} \sum_{(v_i, v_j) \in \mathcal{E}^T} -w_{i,j} \\ \text{s.t.} \quad & \mathcal{T}_m = \mathcal{T}^c_w[\mathcal{V}_m] \quad \text{is connected} \quad \forall m = 1, \ldots, M \end{aligned} \quad (11)$$

Without the sub-group connectivity constraints in (11), the optimal solution corresponds to the Minimal Spanning Tree (MST) among the set of $\mathcal{T}_w^c \in \mathcal{G}$ by definition. To obtain the optimal solution with sub-group connectivity constraints, we propose to define another class of spanning trees as follows and relate its MST to the solution of the problem in (11).

Definition 1. *Given a weighted spanning tree $\mathcal{T}_w^c = (\mathcal{V}, \mathcal{E}^T, \mathcal{W}^T)$ and for all edges $(v_i, v_j) \in \mathcal{E}$ on original graph $\mathcal{G}$, redefine their weights by the following.*

$$w'_{i,j} = \begin{cases} \lambda \cdot w_{i,j}, & \text{if } v_i \text{ and } v_j \text{ are in different sub-groups} \\ w_{i,j}, & \text{else} \end{cases} \tag{12}$$

where $\lambda \in \{\lambda \gg 1 : \lambda \cdot w_{i,j} \gg w_{i',j'}, \forall v_i, v'_i, v_j, v'_j \in \mathcal{V}\}$ is a user-defined constant. Then we call the redefined spanning tree $\mathcal{T}_w^{c'} = (\mathcal{V}, \mathcal{E}^T, \mathcal{W}^{T'})$ as the Connectivity Constraint Spanning Tree (CCST).

Lemma 2. *Define a sub-MST to be a fragment. Let e_{min} be a minimum-weight outgoing edge (MWOE) of a fragment. Connecting e_{min} and its adjacent node in a different fragment yields another fragment in MST, which iteratively will finally output a single MST.*

The proof of Lemma 2 can be found in both [14] and [17].

Lemma 3. *By following the process in Lemma 2 on $\mathcal{T}_w^c$, all nodes within the same sub-group will form a MST fragment before connecting to other sub-group.*

Proof: We prove by contradiction. Suppose the node v_i from sub-group $G(i)$ connect with node v_j first which belongs to sub-group $G(j)$, $G(i) \neq G(j)$. From Lemma 2 we know that at any iteration of connecting MST fragments, they always connect with their minimum-weighted outgoing edge first. Let $v_{i'} \in G(i)$ where there exists an edge between v_i and $v_{i'}$, then from Lemma 2 we know that the weight $w_{i,j} < w_{i,i'}$. This contradicts with the property of $\mathcal{T}_w^c$ in Equation 12. □

Theorem 4. *Given the redefined Connectivity Constraint Spanning Tree (CCST) $\mathcal{T}_w^{c'} = (\mathcal{V}, \mathcal{E}^T, \mathcal{W}^{T'})$ in Definition 1 and denote minimum weight CCST as $\bar{\mathcal{T}}_w^{c'} = \arg\min_{\mathcal{T}_w^{c'} \in \mathcal{T}} \sum_{(v_i,v_j)\in\mathcal{E}^T} -w'_{i,j}$, we have: $\bar{\mathcal{T}}_w^{c'} = \mathcal{G}^{c*}$ in Equation (11). Namely, the Minimum Spanning Tree $\bar{\mathcal{T}}_w^{c'}$ of $\mathcal{G}$ with updated weights (12) is the optimal solution of $\mathcal{G}^{c*}$ in (11) and we call the graph $\bar{\mathcal{T}}_w^{c'}$ as Minimum Connectivity Constraint Spanning Tree (MCCST) of the original connected robotic team graph $\mathcal{G}$.*

Proof: From Lemma 3 we know that each sub-group will be connected as a MST fragment before moving to sub-group edges. By definition, MST within a sub-group $G(i)$ is optimal that gives the minimum total weight, which means

$$\bar{\mathcal{T}}_w^{c'}(i) = \underset{\mathcal{T}_w^{c'}(i)\in\mathcal{T}(i)}{\arg\min} \sum_{(v_i,v_j)\in\mathcal{E}^{T(i)}} -w'_{i,j} \tag{13}$$

Then for the next step, connecting the minimum-weighted outgoing edge between different sub-groups, yields

$$\begin{aligned} \bar{\mathcal{T}}_w^{c'} &= \underset{\mathcal{T}_w^{c'}\in\mathcal{T}}{\arg\min} \sum_{(v_i,v_j)\in\mathcal{E}^{T(i)}} -\lambda \cdot w'_{i,j} \\ &= \underset{\mathcal{T}_w^{c'}\in\mathcal{T}}{\arg\min}\, \lambda \cdot \sum_{(v_i,v_j)\in\mathcal{E}^{T}} -w'_{i,j} = \underset{\mathcal{T}_w^{c'}\in\mathcal{T}}{\arg\min} \sum_{(v_i,v_j)\in\mathcal{E}^{T}} -w'_{i,j} \end{aligned} \tag{14}$$

The equality holds since $\lambda > 0$. □

In this way, we relax the constrained MST optimization problem in (11) into unconstrained MST problem with the same optimality guarantee that can be solved in a principle manner.

B. Construction of Distributed Minimum Connectivity Constraint Spanning Tree (MCCST)

We propose a distributed construction of MCCST of $\mathcal{G}$. For our problem setting, the topology and weights could change every time step, thus a time-optimal real-time algorithm is needed. We develop our algorithm based on the work from [18] and [17]. Different from most of the network algorithms [18][17], our algorithm does not require synchronization, which also reduces the total time. We adapt the method from the previous section. Note that MST is unique for a graph with unique edge weights. Therefore the result is the same from centralized and decentralized construction.

A detailed description of the algorithm is as follows:

1) Overview: The general idea is that the whole system starts as each node forms a individual fragment by itself. All fragments find the *minimum-weight outgoing edge (MWOE)* and join fragments together. Incrementally the forest of fragments will join as a whole tree, that is the MCCST.

Algorithm 1 Distributed MCCST Construction

Input: a: adjacency edge weight list of the original weighted graph
Output: edge list of MCCST

```
1: function ConstructDistributedMCCST(A)
2:     A ← empty adjacency matrix
3:     A ← updated from input a
4:     while msg ← getNewMessage(message_pool) do
5:         if not initialized then
6:             A ← initialRound(msg, A)
7:         else
8:             A ← process(msg, A)
9:         if isConnected(A) then
10:            return getEdgeList(A)
11:        if isEmpty(message_pool) then
12:            resetRound()
```

As shown in Algorithm 1, each robot takes an input of neighboring edge weights and connectivity information, and outputs the computed MCCST edge list. The incoming message is processed according to whether the node is being initialized or not. The process is being reset when there is no new message in the message pool, which implies all the fragments finish updating within themselves and new MWOE need to be connected and a new round begins.

In the initial round, each fragment initially only contains one vertex. Each node directly connect with the neighbor with MWOE. However, to keep information within a fragment consistent, the node with the smallest id is selected

as fragment leader, to keep the computation light-weight. Information keeps updating within the fragment until every node has the same adjacency matrix of the fragment tree.

2) Processing Round: At each processing round, the fragment leader will determine the minimum-weight outgoing edge (MWOE) in its fragment after receiving all MWOE from each fragment node (including itself). Since each node in the fragment only has the local knowledge within its own fragment, it will ask the MWOE neighbor for their fragment information (adjacency matrix, leader id). Whenever a node receives a request to give information, it will reply accordingly. Once each node receives information of MWOE, it will report to the fragment leader. All connect requests will be accepted and this, by lemma 2, always yields a fragment. When the new connection is made, the two fragments will merge their information and update all the nodes.

Algorithm 2 Processing Round of MCCST Construction

Input: msg: incoming messages, A: current adjacency matrix
Output: A: updated adjacency matrix

function PROCESSROUND(msg, A)
  if $leader\_id$ is $self\_id$ **then**
    $MWOE\_cache\_list \leftarrow$ wait($fragment\_node$)
    $MWOE \leftarrow$ min($MWOE\_cache\_list$)
    if all $fragment\_node$ reported **then**
      inform the one with $MWOE$ to connect
  if not reported to leader **then**
    if no MWOE info **then**
      $msg\_pool[MWOE] \leftarrow get\_info$ message
    if get information from MWOE neighbor **then**
      $msg\_pool[leader\_id] \leftarrow report$ message
  $A \leftarrow$ update with msg
  return A

The speed of convergence is highly dependent on the original graph topology and edge weights. The convergence may happen within one single iteration if the MWOE of all fragments directs to a fragment with smaller id. However, if MWOE of neighboring fragment overlaps , it might take $O(\log n)$ iterations to converge and agree on a final MCCST.

Once the final MCCST is obtained as the optimal connectivity graph $\mathcal{G}^c = \mathcal{G}^{c*}$ in (9), we can specify the safety and connectivity barrier certificates (3),(6) to invoke linear constraints and solve the original problem in (7).

V. RESULTS

We evaluate our proposed distributed MCCST method for behavior mixing in a robotic team of 40 mobile robots in Figure 1. The robots are tasked with 4 parallel behaviors and each sub-group either rendezvous (blue and red robots) to its designated task regions or moves to the region and forms a circle (green and magenta robots). As shown in Figure 1a-c, our MCCST approach is able to generate dynamic minimum connectivity graph (red edges) from the present connectivity graph (grey edges) to be minimally invasive to the primary controllers and achieve most of the robot target positions as shown in Figure 1c. In the intermediate stage shown in Figure 1b, although the robots get cluttered around task 4 area, they are able to generate minimum connectivity graph (red links) from the dense local connectivity graph (grey) so that the team can successfully "unfold" and keep moving to their primary targets. In comparison, we present converged results of two other methods with static connectivity graph: Figure 1e always preserves initial MST; Figure 1d and f preserve initial connectivity graph. As the graph is fixed as the robots move, they can hardly achieve circle formation (Figure 1e) or fall into deadlock (Figure 1f). Numeral results are provided in Figure 2 showing our method ensures safety and connectivity, while having minimal control perturbation due to connectivity and maximum task performance (very close to designated target area shown in Figure 2d).

For quantitative results, we run experiments with up to 50 robots and 4 parallel behaviors (four sub-groups simultaneously rendezvous to four different places with safety and connectivity constraints) on Intel Core i7 CPU with 4 cores of 2.70 GHz. For Figure 3a and 3b, the experiment is done by computing the distributed MST 1500 to 3000 times. The complexity of the worst case for the message passing is $O(n \log n)$. However, the average case, as shown in the figure, is better than $O(n \log n)$. Figure 3b shows the average time for computing the distributed MST, which shows that computing distributed MST could be done in real time.

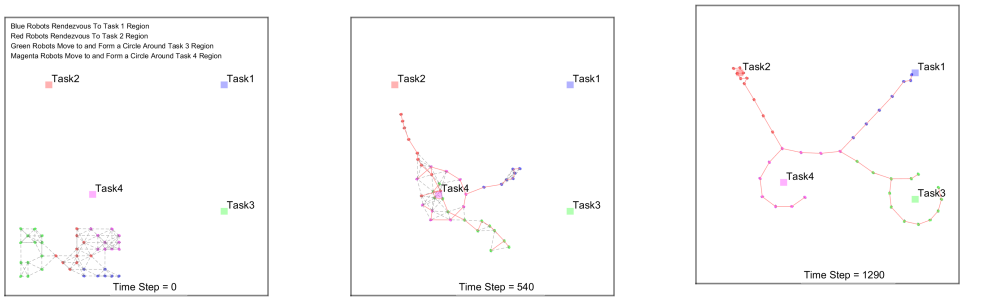
The average distance to target region and perturbation after convergence is calculated from 10 runs with up to 50 robots. In Figure 3c, the average distance to target with MCCST is significantly smaller (closer to target region) than with fixed connection . The distance also decreases as the number of robots increases. However, for the other two methods, the average distance increases with the number of robots, since they are more constrained by unnecessary connections. Figure 3d shows the result of average perturbation. Our method gives much smaller perturbation on average. Note that the result from preserving initial MST gives much worse result than the other two, because the MST edge gives huge deviation from the optimal control output, while the full connection graph gives larger number of constrain edges so that some are canceled out with each other, and the MCCST gives smaller constraints on each tree edge.

VI. CONCLUSIONS

We presented behavior mixing: a framework that enables a swarm in a single connected component to simultaneously perform multiple sequences of behaviors while maintaining safety and connectivity. The robotic team is able to compute in a distributed manner flexible minimum connectivity constraints to ensure that they are minimally invasive to the primary controller with optimality guarantee. This is the first framework to have multiple behaviors executing in parallel. Experiments show that our method is scalable to large number of robots and also significantly improves the swarm performance on a variety of tasks requiring different behaviors.

REFERENCES

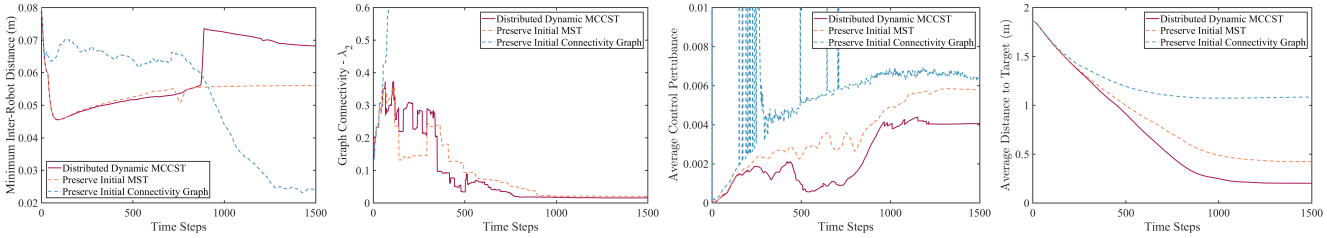
[1] G. Arpino, K. Morris, S. Nagavalli, and K. Sycara, "Using information invariants to compare swarm algorithms and general multi-robot algorithms: A technical report," *arXiv preprint arXiv:1802.08995*, 2018.



(a) Time Step $= 0$ (b) Time Step $= 540$ (MCCST) (c) Time Step $= 1290$ (MCCST, Converged)

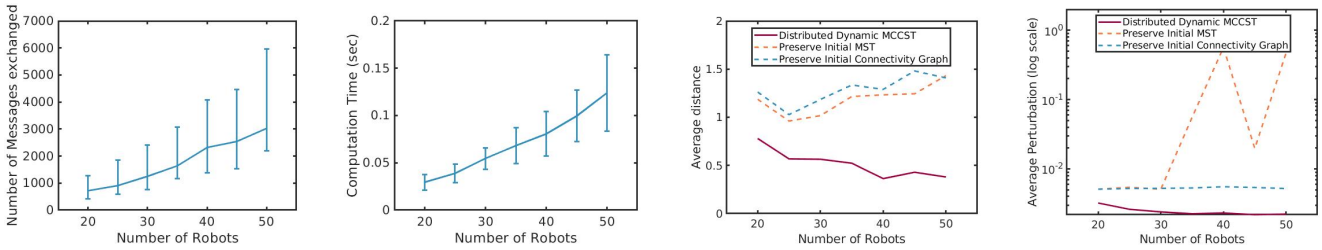
(d) Time Step $= 0$ (e) Time Step $= 1290$ (Fixed Initial MST, Converged) (f) Time Step $= 1290$ (Fixed Initial Connectivity, Converged)

Fig. 1: Simulation example of 40 robots tasked to four different places with behaviour mixing: blue robots and red robots rendezvous to regions of blue task 1 and red task 2 respectively, while green robots and magenta robots move to region of green task 3 and magenta task 4 and form a circle around the regions. Grey dashed lines in (a),(b),(d) denote current connectivity edges and red lines in (a)-(f) denote current active connectivity graph invoking pair-wise control constraints. Compared to fixed inter-robot connectivity constraints from initial MST (e) and initial connectivity graph (f), our proposed MCCST approach (c) enables maximum task performance due to invoked minimum connectivity constraints on the robots.



(a) Minimum inter-robot distance (b) Algebraic connectivity (c) Average control perturbation (d) Average distance to target region

Fig. 2: Performance comparison of simulation example in Figure 1 w.r.t. different metrics: (a) Minimum inter-robot distance (safety distance is 0.02m), (b) Algebraic connectivity evaluated by second smallest eigenvalue of mutli-robot laplacian matrix. Positive meaning connectivity ensured, (c) Control perturbation computed by $\frac{1}{N}\sum_{i=1}^{N}\left\|u_i^* - \hat{u}_i\right\|^2$, (d) Average distance between robots to tasked region (the smaller the better).



(a) Number of messages exchanged (b) Computation Time (c) Average Distance to targets (d) Average Control Perturbation

Fig. 3: Quantitative results summary. (a)-(b) are results from our proposed Distributed MCCST approach. (c)-(d) are comparison results with ours and the other two approaches with static connectivity graph but the same controller (7). (a) Number of messages exchanged during the distribute MST construction. The error bar shows the maximum and minimum number of messages exchanged. (b) Time of constructing the distributed MST. The error bar shows the standard deviation. (c) Average distance from robot to target location after converged. (d) Average control perturbation.

[2] S. Nagavalli, N. Chakraborty, and K. Sycara, "Automated sequencing of swarm behaviors for supervisory control of robotic swarms," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2674–2681.

[3] S. Waharte, N. Trigoni, and S. Julier, "Coordinated search with a swarm of uavs," in *2009 6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops*. IEEE, 2009, pp. 1–3.

[4] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 1128–1134.

[5] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human interaction with robot swarms: A survey," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 9–26, 2016.

[6] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on automatic control*, vol. 51, no. 3, pp. 401–420, 2006.

[7] U. Borrmann, L. Wang, A. D. Ames, and M. B. Egerstedt, "Control barrier certificates for safe swarm behavior." Georgia Institute of Technology, 2015.

[8] L. Wang, A. D. Ames, and M. Egerstedt, "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 2659–2664.

[9] A. Li, L. Wang, P. Pierpaoli, and M. Egerstedt, "Formally correct composition of coordinated behaviors using control barrier certificates," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*. IEEE, 2018.

[10] G. Antonelli, F. Arrichiello, and S. Chiaverini, "Flocking for multi-robot systems via the null-space-based behavioral control," *Swarm Intelligence*, vol. 4, no. 1, p. 37, 2010.

[11] M. M. Zavlanos, A. Jadbabaie, and G. J. Pappas, "Flocking while preserving network connectivity," in *Decision and Control, 2007 46th IEEE Conference on*. IEEE, 2007, pp. 2919–2924.

[12] X. Yan, J. Chen, and D. Sun, "Multilevel-based topology design and shape control of robot swarms," *Automatica*, vol. 48, no. 12, pp. 3122–3127, 2012.

[13] L. Sabattini, N. Chopra, and C. Secchi, "Decentralized connectivity maintenance for cooperative control of mobile robotic systems," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1411–1423, 2013.

[14] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and systems (TOPLAS)*, vol. 5, no. 1, pp. 66–77, 1983.

[15] J. Nešetřil, E. Milková, and H. Nešetřilová, "Otakar boruvka on minimum spanning tree problem translation of both the 1926 papers, comments, history," *Discrete mathematics*, vol. 233, no. 1-3, pp. 3–36, 2001.

[16] A. D. Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer, "Distributed verification and hardness of distributed approximation," *SIAM Journal on Computing*, vol. 41, no. 5, pp. 1235–1265, 2012.

[17] D. Peleg, "Distributed computing," *SIAM Monographs on discrete mathematics and applications*, vol. 5, 2000.

[18] G. Pandurangan, P. Robinson, and M. Scquizzato, "A time-and message-optimal distributed algorithm for minimum spanning trees," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2017, pp. 743–756.

[19] M. Elkin, "A faster distributed protocol for constructing a minimum spanning tree," *Journal of Computer and System Sciences*, vol. 72, no. 8, pp. 1282–1308, 2006.