Synthesis of Controllers to Create, Maintain, and Reconfigure Robot Formations with Communication Constraints

Nora Ayanian, Vijay Kumar, and Daniel Koditschek

Abstract We address the synthesis of controllers for groups of multi-robot systems that enable them to create desired labelled formations and maintain those formations while travelling through an environment with obstacles, with constraints on communication. We assume that individuals in a group are capable of close coordination via high bandwidth communication, but coordination across groups must be limited because communication links are either sporadic or more expensive. We describe a method for developing feedback controllers that is entirely automatic, and provably correct by construction. We provide a framework with which navigation of multiple groups in environments with obstacles is possible. Our framework enables scaling to many groups of robots. While our paper mainly addresses groups of planar robots in $\mathbb{R}^2$, the basic ideas are extensible to $\mathbb{R}^3$.

1 Introduction

There are many types of tasks which require multiple groups working on subtasks concurrently. For example, building an automobile requires many subassemblies which are built in parallel for efficiency. When building a house, each wall of the wooden frame is built separately, then fastened together. These kinds of tasks require coordination of groups on different levels: we want each group to work closely to accomplish subtasks, but we also want them to coordinate on the inter-group level to ensure that they accomplish the higher-level task.

These types of tasks often occur in cluttered environments, where it may be costly (either in time, energy, or other currency) to allow groups to communicate large amounts of information to other groups. Exchanging information with unnecessary detail results in loss of time and energy, thereby increasing cost. It is imperative to

Nora Ayanian · Vijay Kumar · Daniel Koditschek
GRASP Laboratory, Philadelphia, Pennsylvania,
e-mail: {nayanian, kumar, kod}@grasp.upenn.edu
We gratefully acknowledge the support of NSF grants IIS-0413138 and IIS-0427313, ARO grant W911NF-05-1-0219, ONR grants N00014-07-1-0829 and N00014-08-1-0696.
N. Ayanian gratefully acknowledges support from the NSF.

exchange information between groups on a limited basis, while ensuring that ultimately the task is accomplished. We address the problem of synthesizing controllers for labelled robots working in multiple groups in the same workspace, merging and splitting to form a group of desired size and formation shape for a specific task. We will assume that individuals in a group are capable of close coordination via high bandwidth communication but coordination across groups has to be limited because communication links are either sporadic or more expensive.

The configuration space for an N robot system is the Cartesian product of each robot's configuration space, $\mathscr{C} = \mathscr{C}_1 \times \mathscr{C}_2 \times \cdots \times \mathscr{C}_N$. Planning in a space of such large dimension causes computational as well as combinatorial problems which are challenging to address. To decrease complexity, constraints on desired inter-robot distances or relative positions are typically added, reducing dimensionality. Instead, we propose an approach which allows more flexibility. Our goal is to construct controllers similar to navigation functions, which have desirable properties such as safety and convergence guarantees [15], while allowing for larger group sizes.

1.1 Related Work

Similar problems have been studied in the mobile robotics literature both in large and small groups of robots. In small sized groups we can provide guarantees and formal proofs that specific formations can be achieved and maintained, even in the presence of obstacles [2,4,5,14]. As the groups grow, however, formal proofs [4] or controller synthesis [2,15] become prohibitively complex.

In large groups, one cannot feasibly synthesize a specialized controller for each robot, thus achieving specific, labelled formations is not addressed. Some works use abstractions to control an entire group [3, 12, 19]; in this case, navigation and obstacle avoidance is at the abstraction level, decreasing computation significantly. However, we forfeit control over the network topology, which can change as the group moves. In [3], safety is not guaranteed: robots can collide and escape from the abstraction. Some limitations of [3] are addressed in [12], which still does not enable us to specify formations in the sense of exact shape and topology. A particular formation can be sepcified in [19], but the number of moments which must be supplied to specify a particular formation increases with the number of robots, and the method is not entirely automatic.

Flocking or schooling strategies also enable control of large groups of robots with relatively little computation [18]. These strategies stabilize the entire group's velocity to a single velocity. However, like the above large scale controllers, they lack the capability of specifying particular formations; the final shape of the formation depends on the initial conditions, and cannot be controlled directly.

In [7], large groups of robots are stabilized to shapes. However, the presence of obstacles in the workspace could cause local minima or deadlock. Additionally, there is no ordering to the robots on the shape; this is a function of initial conditions.

With sizes between small and large groups, one can provide some guarantees while taking advantage of some reduction in complexity. In [13], proofs are provided for creating and maintaining formations, but collision avoidance is guaranteed only in most cases, requiring careful parameter choices. In [10], undesirable local minima

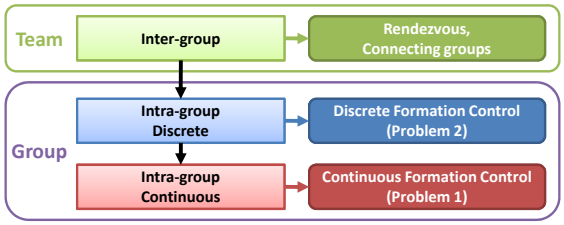


Fig. 1 The levels of hierarchy in our controller. We address group navigation at the team level, decoupling it from the formation problem. This allows us to limit the complexity of the problem.

can occur if sufficient virtual leaders are not added. Additionally, the authors do not provide a stable way to switch between formations. In [17] a method is proposed for creating a formation and maintaining it during motion. However, no guarantees are made in the presence of obstacles, and formations must be unlabelled.

1.2 Proposed Approach

The method we propose is for large but finite-sized *groups* with labels; it provides global guarantees on shapes, communication topology, and relative positions of individual robots. We define a *group* of agents as a collection of agents which work simultaneously to complete a single task. Two or more groups act in a *team* to complete a task which requires completing multiple parallel subtasks [1]. Our contribution is twofold. First, we provide a framework for synthesizing controllers for multiple groups of robots in environments with obstacles with communication constraints. Second, we provide a method for *automatically* reconfiguring groups of robots into desired labelled formation shapes without local minima.

A key feature in our approach is a hierarchical decomposition of the problem of constructing feedback controllers (Fig. 1). This reduces the complexity of synthesizing individual robot controllers that meet such specifications as collision avoidance and shapes of formations. We design multi-group navigation controllers at the team level, and the control input for individual robots is derived by summing the inputs from the feedback controller for the robot within its group and the feedback controller for the group. The multi-group navigation problem is equivalent to the multi-robot navigation problem, so we focus in this paper on merging groups of robots into groups of arbitrary numbers of robots, and constructing desired formation shapes.

Our approach is as follows. First, the groups which are involved in the merge convene at some negotiated rendezvous area. We abstract the groups by enclosing them in deformable rectangles, centered at the group centroid and sized appropriately (we address size in Section 4). When the groups are within a pre-specified merging distance, they merge into one group and begin reconfiguration, ending at the desired formation shape. In the case that there are more robots than required for the task assignment, the extra robots split into a separate group. Once the desired formation is achieved, the newly-formed group(s) navigates toward the task.

In Section 2 we formulate the problem. In Section 3 we develop controllers for reconfiguration and formation maintenance. In Section 4 we describe and develop controllers on the abstraction. In Section 5 we describe the process of merging and splitting groups, and follow in Section 6 with MATLAB simulation results. We discuss complexity in Section 7, and conclude in Section 8.

2 Problem Formulation

Consider a team with multiple groups, $\mathscr{G}^i, i = \{1,\ldots,m\}$, of N^i kinematic agents $V_A^i = \{a_j^i | j = 1,\cdots,N^i\}$. A group consists of a small number of robots, which can communicate with each other at high bandwidth, enabling centralization. While communication is facilitated by having a complete graph, it is not necessary since agents can exchange information about their neighbors. The team must form a group of $N^g \leq \sum_{i=1}^{m} N^i$ agents to accomplish a large task. Each agent has the configuration or state $x_j^i \in \mathbb{R}^2$ with the dynamics:

$$\dot{x}_j^i = U_j^i,\ x_j^i \in X_j^i \subset \mathbb{R}^2,\ j = 1,\ldots,N^i. \tag{1}$$

Within groups, communication of state information occurs very frequently, so that control is centralized over the entire group. Communication across teams occurs less frequently. Long-range communication occurs rarely if at all, requiring decentralized control.

We use an abstraction on groups of robots to reduce the computational complexity of the problem.

Definition 1. An *abstraction* of a group of robots is a surjection

$$S : \mathscr{C}_T^i \to B,\ S(x^i) = b \tag{2}$$

so that the dimension of B is not dependent on the number of robots N^i.

The abstraction models the extent of the formation, which we call the *boundary*, while the controllers on the robot level ensure that the boundary is satisfied. Therefore, a group of robots can reconfigure from one formation to another knowing only the limits of the abstraction, decoupling the agents from the physical space. The specific abstraction we use is discussed in detail in Section 4.

Figure 2 is a graphical representation of the hierarchichal structure of our approach. At the top level, groups interact with limited knowledge of other groups. At the middle level, there is interaction between individual robots in order to maintain the formation. At the lowest level, individual robots execute the continuous controller. In the examples we present, we assume that there are no obstacles within the group boundary. However, should an obstacle appear within a group boundary, the group can split appropriately, then rejoin in a location without obstacles.

The input to each agent is the sum

$$U_j^i = u_j^i + u_b^i \tag{3}$$

where u_j^i is the formation shape controller (Section 3), and $u_b^i \in \mathbb{R}^2$ is the abstraction controller (Section 4). As shown in Fig. 2, the two control inputs drive dynamics at two different time scales. Group dynamics (motion within a group) must evolve on a much faster time scale than the team dynamics (motion of the group). In other words, time required for robots to converge to a target formation within the group is much smaller than the motion of the group. This time-scale separation is necessary to guarantee convergence at all levels.

The number of agents required for the task, the goal formation shape, and the environment are known to all agents. We assume each agent is capable of synthesizing

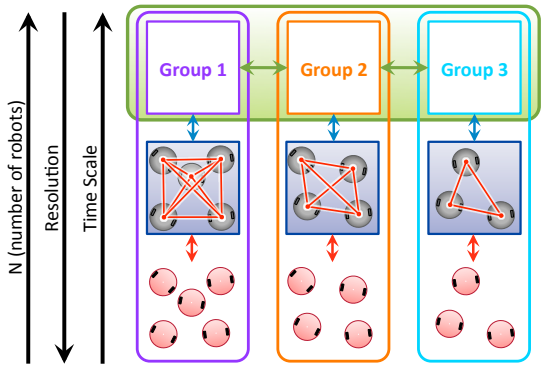


Fig. 2 Hierarchical structure. At the top level, groups interact with limited knowledge about other groups. Within each group, formations must be maintained. At the lowest level, individual robots implement the continuous controllers. As the number of robots increases, the spatial resolution required for planning and control decreases, and the time scale increases so that dynamics get faster.

controllers (both for group navigation and reconfiguration), its group's abstraction, and whether certain criteria are satisfied (such merging criteria). This information propagates through the group rapidly through explicit communication, therefore we are not concerned with which agent is responsible for these calculations. Agents observe relative state of their neighbors and exchange this information with other neighbors to construct a complete group configuration. Groups are capable of long-range communication in short bursts to determine a rendezvous point. Once the rendezvous point is determined, they no longer use long-range communication.

3 Formation Shape Controllers

In this section we develop the formation shape controllers, which both reconfigure the robots and maintain the formation once it is achieved.

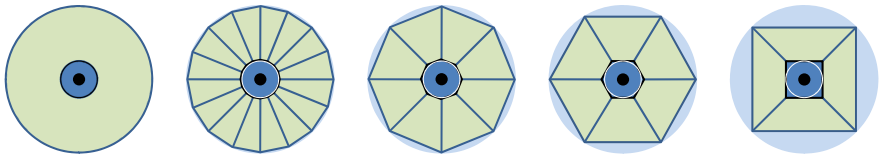
Let the set of all agents be $V_A \equiv V_A^1 \cup V_A^2 \cup \cdots \cup V_A^m$. (Hereafter, for simplicity, where we describe a property for $\cup_{i=1}^{m} \cup_{j=1}^{N^i} a_j^i$, we will drop the superscript i.) Connectivity between all agents V_A is modeled by a *robot formation graph*. Agents must maintain *proximity constraints*, which are represented by edges on the robot formation graph and the collision graph. Recall that a *graph* is a pair of sets $G = (V,E)$, where $V = \{v_1, \ldots, v_n\}$ is the set of vertices or nodes and $E \subseteq [V]^2$ is the set of edges on the graph. Pairs of vertices for which $(v_i, v_j) \in E$ are called adjacent. A graph in which all pairs of vertices are adjacent is called a complete graph.

Definition 2. A *robot formation graph* is a graph $G_N^\rho = (V_A, E_N)$ where E_N is the set of edges which denote pairs of agents which directly communicate state information, and ρ is a metric for determining inter-agent distances. To enable communication, pairs $(a_j, a_k) \in E_N$ must be within a maximum distance $|x_j - x_k|_\rho \leq \delta_{max}$. The constraint can be written

$$v^\rho(x_j, x_k) \leq 0 \;\; \forall (x_j, x_j) \in E_N. \tag{4}$$

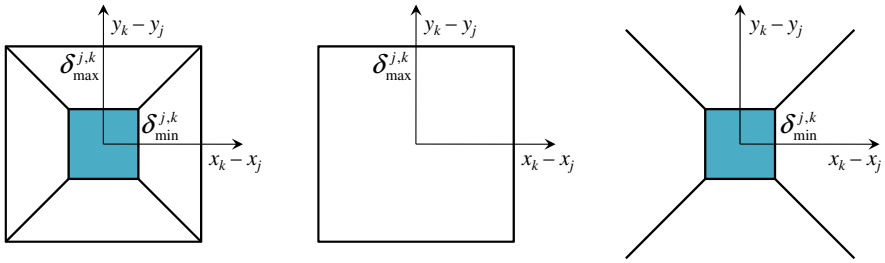
We call pairs of agents which are adjacent on this graph *neighbors*.

Definition 3. The *collision graph* on a group $\mathscr{G}_i$ of agents is a static graph $G_L^{i,\rho} = (V_A^i, E_L^i)$ where E_L^i is the set of all pairs of agents in V_A^i which cannot occupy the same coordinates simultaneously. Pairs $(a_j^i, a_k^i) \in E_L^i$ must maintain a nonzero minimum



(a) (b) (c) (d) (e)

Fig. 3: The annulus in relative space of a pair of robots (a) and a few approximations (b)-(e). The dark blue center corresponds to the collision constraint, while the light green area depicts allowable configurations. Note that in panels (b)-(e) the dark blue polygons are overapproximations of the unsafe region (encircled in white), while the large green polygons underapproximate the safe region (shaded blue circle).



(a) (b) (c)

Fig. 4: Proximity constraints for pairs of robots. Shaded area indicates illegal configurations. (a) Neighbors with collision constraints. (b) Neighbors with no collision constraint. (c) Collision constraint on non-neighbors.

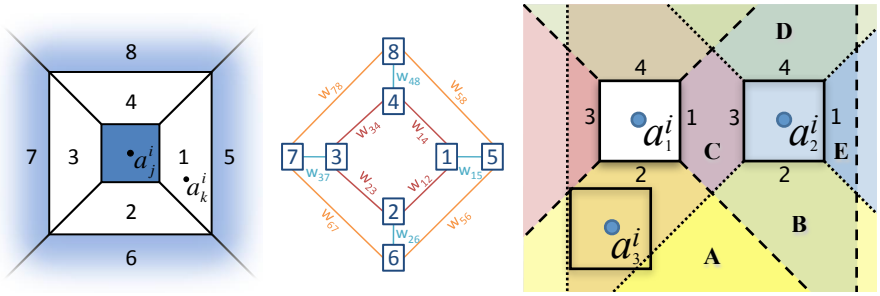
distance $|x^i_j - x^i_k|_\rho \geq \delta_{min}$. The constraint can be written

$$\lambda^{i,\rho}(x^i_j, x^i_k) \geq 0 \;\; \forall (x^i_j, x^i_k) \in E^i_L. \tag{5}$$

For homogeneous agents occupying the same space, this graph will be complete.

The proximity constraints specified by the robot formation graph $G^\rho_N = (V_A, E_N)$ and the collision graph $G^{i,\rho}_L = (V^i_A, E^i_L)$ are realized using metric ρ. For pairs of agents $(a^i_j, a^i_k) \in E_N \cap E^i_L$ the intersection of these constraints corresponds to an annulus in the relative space of two agents. One can choose any underestimation of the annulus, with a tessellation consisting of convex regions, and call this the metric ρ. A few options are shown in Fig. 3. Our metric of choice is the infinity norm, in Fig. 3e since it has fewer regions, decreasing complexity and allowing more flexibility in the formation. Hence, the proximity constraints become a square annulus in the relative space of two agents as in Fig. 4a (the shaded region denotes illegal configurations). Pairs $(a^i_j, a^i_k) \in E_N - E^i_L$ have only the maximum distance constraint (Fig. 4b), and pairs $(a_i, a_j) \in E_L - E_N$ have infinite annuli (Fig. 4c).

To accomplish the task, a specific formation shape is required of the new group. The formation shape can be provided in exact, continuous form, or approximate, discrete form. In defining the discrete robot formation shape, we desire a non-



(a) (b) (c)

Fig. 5: The shape descriptors (regions) for discrete robot formation shapes. (a) For pair (a^i_j, a^i_k) the shape descriptor represents the location of a^i_k with respect to a^i_j. Here, the region is 1. (b) We can use this discrete system to build an adjacency graph. (c) Overlapping proximity regions of a group of three robots. Dashed (dotted) lines correspond to boundaries of proximity regions for a^i_1 (a^i_2). Letters A-E correspond to possible polytopes through which a^i_3 would pass to get to $\mathscr{F}^i_d = (1,1,1)$.

overlapping partition of the square annulus whose union is the entire region. The description must also contain information about connectivity.

Definition 4. A *robot formation shape* $\mathscr{F}$ of N robots describes the relative locations of the set of robots, specified exactly using continuous shape variables. $\mathscr{F}$ is a $\binom{N}{2}$-tuple of vectors $\mathscr{F} = \{r_{(1,2)}, r_{(1,3)}, \ldots, r_{(N-1,N)}\}$ where $r_{(j,k)} = x_k - x_j$. We use a superscript ($\mathscr{F}^i$) to refer to the subset corresponding to group $\mathscr{G}^i$.

Definition 5. The *discrete robot formation shape* $\mathscr{F}_d$ of N robots describes approximate relative locations of the set of robots using discrete shape descriptors. The shape descriptors are integers corresponding to regions of the tessellation of the annulus. $\mathscr{F}_d$ is a $\binom{N}{2}$-tuple of these integers, $\mathscr{F}_d = \{f_{(1,2)}, f_{(1,3)}, \ldots, f_{(N-1,N)}\}$. For each pair of robots (a_j, a_k) $j < k$, $f_{(j,k)}$ describes the position of a_k with respect to a_j, according to Fig. 5a. Regions 1-4 correspond to communication between the pair (i.e. $(a_j, a_k) \in E_N$). Regions 5-8 correspond to no direct communication between the pair ($(a_j, a_k) \notin E_N$). We use ($\mathscr{F}^i_d$) to refer to the subset corresponding to group $\mathscr{G}^i$.

An example discrete formation shape for a group of three robots is shown in Fig. 5c. Here, $f^i_{(1,2)} = 1$, $f^i_{(1,3)} = 2$, and $f^i_{(2,3)} = 3$, so that $\mathscr{F}^i_d = (1,2,3)$.

To build the space on which we develop the controllers, we combine the proximity constraints with the configuration spaces of the robots.

3.1 The Task Configuration Space

Definition 6. The *configuration space* $\mathscr{C}^i_j$ of an agent a^i_j is the set of all transformations of a^i_j. The *free space* $\mathscr{C}^{i,free}_j$ of a^i_j is the set of all transformations of a^i_j which do not intersect with obstacles in the configuration space.

In this work, navigation (and therefore obstacles), are dealt with on the abstraction level. The abstraction constrains the free space of a group so that $\mathscr{C}^i_j \neq \mathscr{C}^{i,free}_j$. In fact, $\mathscr{C}^{i,free}_j$ is a local space, whose origin is the center of the abstraction.

Definition 7. The *group configuration space* is the Cartesian product of the free spaces of each agent in a group,

$$\begin{gathered}\mathscr{C}^i_{all} = \mathscr{C}^{i,free}_1 \times \mathscr{C}^{i,free}_2 \times \cdots \times \mathscr{C}^{i,free}_{N^i} \\ x^i = [x^i_1, \ldots, x^i_{N^i}] \in \mathscr{C}^i_{all}.\end{gathered} \tag{6}$$

Thus the configuration of a group $\mathscr{G}^i$ of N^i agents is described by a single point in $\mathscr{C}^i_{all} \subset \mathbb{R}^d$, $d = 2N^i$.

Definition 8. The *task configuration space* $\mathscr{C}^i_T$ for the group $\mathscr{G}^i$ is the set

$$\begin{aligned}\mathscr{C}^i_T &= \mathscr{C}^i_{all} \cap \mathscr{L}^i_\rho \cap \mathscr{N}_\rho, \\ \mathscr{L}^i_\rho &\equiv \{x^i | x^i \in \mathscr{C}^i_{all}, \lambda_\rho(x^i_j, x^i_k) \geq 0 \ \forall (a^i_j, a^i_k) \in E^i_L\}, \\ \mathscr{N}_\rho &\equiv \{x^i | x^i \in \mathscr{C}^i_{all}, \nu_\rho(x^i_j, x^i_k) \leq 0 \ \forall (a^i_j, a^i_k) \in E_N\}.\end{aligned} \tag{7}$$

$\mathscr{C}^i_T$ is a polytopic space in which robots cannot collide or lose communication.

Note that each group's task configuration space is independent of other groups; that is, robots in a group rely only on each other's positions. Thus, reconfigurations of a group from one formation to another is entirely decoupled from other groups.

Each discrete group formation corresponds to a unique polytope in $\mathscr{C}^i_T$. By planning and synthesizing controllers on $\mathscr{C}^i_T$, we drive the robots to the desired formation and keep them there as the group navigates the space.

Problem 1. For any initial group formation $\mathscr{F}^i_0$, consider the system (1) on $\mathbb{R}^{2N^i}$, with goal formation $\mathscr{F}^i_g$ and metric ρ. Find an input function $u^i : [0, T_0] \to \mathscr{U}$ for any $x^i_0 \in \mathscr{F}^i_0 \subset \mathscr{C}^i_T$ such that

1. for all time $t \in [0, T_0]$, $x^i \in \mathscr{C}^i_T$ and $\mathscr{F}^i_{T_0} = \mathscr{F}^i_g$,
2. $\dot{x}^i_j = u^i_j$,
3. $x^i(t) \in \mathscr{L}^i_\rho \cap \mathscr{N}_\rho$, $\forall t \in [0, T_0]$.

3.2 Shape Controllers on $\mathscr{C}^i_T$

In this section, we synthesize feedback controllers to solve Problem 1. We follow closely [2], specifying our modifications for the present problem. We choose to use a centralized version, since state information is shared among all connected agents. Unlike [2], we simultaneously build a discrete representation of the task configuration space and find a path in this representation. We then translate the path into feedback controllers. The key step in the first stage is to define an adjacency graph on the set of polytopes.

Definition 9. The *polytope graph* $G_P^i = (V_P^i, E_P^i)$ on the polytopes in $\mathscr{C}_T^i$ is the pair of sets $V_P^i = \{c_1^i, \ldots, c_n^i\}$, where c_q^i is the Chebyshev[1] center of the q-th polytope P_q^i, and E_P^i, the set of all pairs of polytopes which share a facet.

By using the discrete group formation shape notation, we build the polytope graph online, on an as-needed basis. For example, in the three robot group formation $\mathscr{F}_d^i = (1,2,3)$ in Fig. 5c, we use the graph in Fig. 5b to generate adjacent formations by moving to adjacent regions. By changing the region corresponding to the location of a_2^i with respect to a_1^i, we get $\mathscr{F}_d^i = (2,2,3)$ and $\mathscr{F}_d^i = (4,2,3)$. By changing the integer corresponding to the location of a_3^i with respect to a_1^i, we get $\mathscr{F}_d^i = (1,1,3)$ or $\mathscr{F}_d^i = (1,3,3)$. Using this method, we build an adjacency graph online while concurrently minimizing cost via a graph search algorithm.

In Fig. 5b we have included separate weights w_{pq} for each edge. Thus the cost of moving from one formation[2] $\mathscr{F}_d^r$ to another adjacent formation $\mathscr{F}_d^{r+1}$ is

$$C\left(\mathscr{F}_d^r, \mathscr{F}_d^{r+1}\right) = \sum_{j=1}^{N} \sum_{\substack{k=1 \\ k>j}}^{N} w_{f^r_{(j,k)} f^{r+1}_{(j,k)}}. \tag{8}$$

This induces a heuristic cost for going from the initial to the goal formation. If $\mathscr{F}_d^0$ and $\mathscr{F}_d^g$ are the initial and final formations, the minimum cost of reconfiguring is

$$h\left(\mathscr{F}_d^0, \mathscr{F}_d^g\right) = \sum_{j=1}^{N^i} \sum_{\substack{k=1 \\ k>j}}^{N^i} w_{f^0_{(j,k)} f^g_{(j,k)}}.$$

In our simulations, we choose to minimize the number of transitions, and thus set all weights $w_{pq} = 1$.

It is important to note that not all nodes will be valid. From Fig. 5c it is obvious that $\mathscr{F}_d^i = (4,2,3)$ is an invalid formation: it is impossible for a_2^i to be in region 4 of a_1^i and simultaneously have a_3^i in region 3 of a_2^i. Therefore, while we anticipate nodes by using the graph in Fig. 5b, before adding nodes to G_P^i we check if the polytopes are empty, corresponding to invalid formations. Thus the cost heuristic can be used as an underestimate for the cost-to-goal in a best-first-search algorithm.

Problem 2. For the initial discrete group formation shape $\mathscr{F}_d^{i,0}$, find a path $t = \{c_{t^i_1}^i, \cdots, c_{t^i_g}^i\}$ on G_P^i to the goal discrete formation shape $\mathscr{F}_d^{i,g}$, such that we minimize the actual cost

$$h^*\left(\mathscr{F}_d^0, \mathscr{F}_d^g\right) = \sum_{r=0}^{g-1} C\left(\mathscr{F}_d^{i,r}, \mathscr{F}_d^{i,r+1}\right).$$

Theorem 1 (Necessary and Sufficient condition). *Problem 1 has a solution iff Problem 2 has a solution.*

Proof. $\mathscr{C}_T^i$ contains every allowable configuration x^i in our polytopic world model. G_P^i contains all the information about the connectivity of $\mathscr{C}_T^i$. Thus, if there is a

[1] The Chebyshev center of a polytope is the center of the largest inscribed ball.

[2] Although this applies to a group $\mathscr{G}^i$, here we drop the subscript i for clarity.

solution to Problem 1, there must exist a path from the start node in G_P^i to the goal node. Conversely, if there is no path on the graph G_P^i between the start node and the goal node, there is no solution to Problem 1. □

Corollary 1 (Completeness). *Problem 2, and therefore Problem 1, has a solution if the start and goal nodes on the polytope graph G_p^i are connected.*

After a path to the goal is determined, we want to be able to synthesize feedback controllers to drive the system through those polytopes to the goal. We choose to use a controller developed by Lindemann and Lavalle which is applicable in high dimensions and results in smooth feedback [11].

We set the vector field on each facet except the exit facet to be a unit inward normal; on the exit facet, we set the vector field to the unit normal pointing outward. By using a bump function, vector fields on the faces of each transitional polytope are smoothly blended with an attractor field on the Generalized Voronoi Diagram (GVD) of the polytope. For each polytope on the path to the goal formation, we implement the controller using the Chebyshev center of the exit facet as the attractor.

In the goal polytope, if an exact formation shape is prescribed, we decompose the polytope so that the vertices of each polytope in the decomposition are the vertices of a facet along with the goal point. Then we set all facet vector fields to be pointing inward, and the field on the faces of the decomposition always pointing to the goal.

If a discrete formation shape is prescribed, we set the attractor field to always point to the Chebyshev center of the polytope (not the exit facet), satisfying the requirements set in [11] to guarantee convergence. There are several advantages to using the Chebyshev center. First, it allows a minimum radius around the attractor, providing some robustness (for disturbances or feedback linearization for nonholonomic robots). Second, the Chebyshev center lies on the GVD, so we do not need to use a separate decomposition that would force us to enumerate the vertices of the polytope, which is computationally expensive.

To smoothly stabilize the system at the Chebyshev center, we normalize the attractor field until it is within the radius of the Chebyshev ball. Once within the Chebyshev ball, we use a second bump function to drive the input to zero as we approach the center.

We use the controller for the goal polytope to maintain the desired group formation of the group as it moves through the space, inside the bounds of the abstraction.

4 Geometric Abstractions for Groups

The abstraction enables robots to operate in an obstacle-free (but limited) environment, while group navigation is handled by a higher-level controller for the abstraction. This controller is then summed onto the individual controllers as in (3). .

Definition 10. The *group abstraction* B^i is a pair

$$B^i = \left(x_b^i, s^i(N^i)\right) \in \mathbb{R}^4 \tag{9}$$

where $s^i(N^i)$ is a shape vector, and

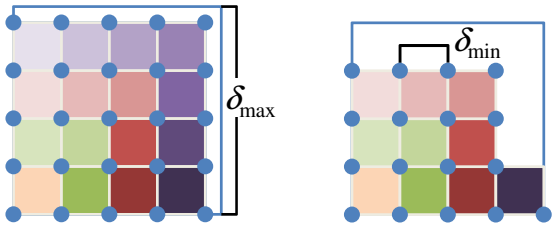


Fig. 6 The limits of the size of the abstraction. (a) The upper bound for number of robots in a group is a function of the ratio $(\delta_{\max}/\delta_{\min})$. (b) The minimum size of the square is a function of $\delta_{\min}$ and N^i.

$$x_b^i = \frac{1}{N^i}\sum_{j=1}^{N^i} x_j^i.$$

The center of the group abstraction is x_b^i, and the shape vector $s^i(N^i)$ represents the boundary and size of the abstraction which encloses the group of robots.

Although any boundary can be chosen for the group, we choose a rectangle for all groups boundaries since a rectangle corresponds well with our choice of the infinity norm. Therefore, the shape vector is a pair $s^i(N^i) = (s_w^i, s_h^i)$ where s_w^i corresponds to the width and s_h^i to the height of the rectangle. Knowing the shape of the abstraction, we can determine the minimum size of the abstraction so that the rectangle is large enough to contain the number of robots in the group,

$$N^i \leq \left(\lfloor s_w^i(N^i)/\delta_{\min}\rfloor + 1\right)\left(\lfloor s_h^i(N^i)/\delta_{\min}\rfloor + 1\right). \tag{10}$$

To ensure graph completeness in the abstraction, we can also enforce $\{s_w^i(N^i), s_h^i(N^i)\} \leq \delta_{\max}$. This induces a limit for the number of robots which can be in a group with a complete graph using our abstraction

$$N^{\max} = \left(\lfloor \delta_{\max}/\delta_{\min}\rfloor + 1\right)^2. \tag{11}$$

An example of the maximum number of robots in a group is shown in Fig. 6a. Here, $\delta_{\max}/\delta_{\min} = 4$, so that $N^{\max} = (4+1)^2 = 25$ is the maximum number of robots in the group. An example of the minimum of $s^i(N^i)$ is in Fig. 6b. Here, $\lfloor\sqrt{N^i}\rfloor = 4$, so the minimum width is $4\delta_{\min}$.

We treat the abstraction as a single robot. Because multiple abstractions share one workspace, we need a multi-robot controller to ensure they do not collide.

To emulate the cost of sharing information across large spaces, we place restrictions on communication between groups of robots on three levels based on inter-group distances. The lowest level of communication occurs at the largest distances, above the threshold $\gamma^{\max}$,

$$|x_b^i - x_b^k|_\rho > \gamma^{\max}.$$

At this level, groups communicate as necessary to negotiate rendezvous points.

The mid-level of inter-group communication occurs below the threshold $\gamma^{\max}$,

$$|x_b^i - x_b^k|_\rho \leq \gamma^{\max},$$

where groups perceive position relative to each other $(x_b^i - x_b^k)$.

Once two groups are close enough that explicit inter-group communication is established (i.e. a member from one group is able to communicate directly with a member of the other group), groups can share both the number and position of each group's agents $(x_j^i, j = 1,\ldots,N^i)$ by passing this information through the robot formation graph. Once the group is within a pre-specified distance of the other groups,

$$|x_b^i - x_b^k|_\rho \leq \gamma^{\min},\ \forall\, i,k \in \{1,\cdots,m\} \tag{12}$$

the groups are able to commence the merging process.

As discussed in Section 1.1, there are many controllers applicable to multi-robot problems. However, the inter-group communication restrictions as well as the real-time nature of this problem require a decentralized controller to bring the groups to the rendezvous area. (In our simulations, we have used path-planning to determine a path for each group to the rendezvous point.) Once the groups are close enough to know relative position $(x_b^i - x_b^k \leq \gamma^{\max})$, a more demanding controller may be used to drive them close enough to communicate and merge (until they satisfy (12)). Then, they must reconfigure into the desired formation, and continue to the task location while maintaining that formation.

5 Merging and Splitting Groups

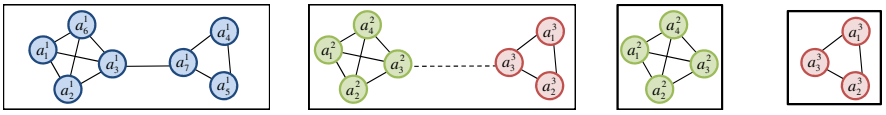
In this section we describe the process of merging groups. Once the groups have satisfied (12), they combine their boundaries into the smallest boundary of the specified shape that contains all of the groups' boxes. This will now be the boundary for the reconfiguration discussed in Sec. 3.2.

The desired formation can be either connected or disconnected. If we have just the right number of robots, the resulting graph is connected. If we have too many, the formation graph is disconnected, and a group of robots break away.

If we have the exact number of robots required for the task, once they have reconfigured into the desired formation, the boundary size must be adjusted. The boundary is resized to within some small ε of the smallest rectangle centered at the centroid of the group and enclosing all the robots. If it is possible to resize directly to the desired size (in the case of a rectangle, $s_w^i(N^i) \times s_h^i(N^i)$), then we are done, and the group can continue to the task. If not, we allow the robots to stabilize to the Chebyshev center of the formation using the new boundary size. Then, we re-evaluate the boundary size, and iterate until we get to the desired size.

If we have more robots than required, the group reconfigures into a formation shape such that the required robots' subgraph is that required for the task, and the rest of the robots are connected to that subgraph by one edge. An example of this is shown in Fig. 7. The dotted line in the example depicts where the group will split. The desired formation is achieved after reconfiguration in Fig. 7a. The discrete formation shape integer relating a_3^1 to a_7^1 is $f_{3,7}^1 = 1$. In Fig. 7b, the groups separate, until $f_{3,7}^1 = 5$, meaning there is no direct communication between a_3^1 and a_7^1. This is when the group splits into two as in Fig. 7c.

In summary, the algorithm for our approach involves the following six steps.



(a) Desired formation shape (b) Preparing to split (c) After splitting

Fig. 7: An example of a desired robot formation shape and the splitting process when the task requires less robots than the total number in all groups.

Algorithm 1

1. *Construct the goal controller for formation maintenance in $\mathscr{C}_T^i$ for each group of robots.*
2. *Drive the groups toward each other in the space.*
3. *When (12) is satisfied, solve Problem 2 while selectively constructing the task configuration space for the joint group of robots.*
4. *Solve Problem 1 on all polytopes on the path, and solve for a goal controller in the goal polytope.*
5. *If $\sum_{i=1}^{m} N^i > N^g$, break the team into two separate groups and construct the task configuration space and goal polytope controller for the new groups.*
6. *Drive the newly formed group(s) to the task location while using the new goal polytope controller to maintain the formation.*

6 Simulations

We simulate a two-group example where the groups merge into the correct number of robots required for the task, and a three-group example where there are more robots than necessary to complete the task. The simulations run on MATLAB, using the Multi-Parametric Toolbox for polytope computations [9].

Although our simulations run on MATLAB, by using a MATLAB interface for PLAYER and GAZEBO, we can transition this controller directly to a three-dimesional dynamic simulator (GAZEBO) or perform experiments on robots using PLAYER, parts of the PLAYER/STAGE/GAZEBO project [6]. As in [2], we can use feedback linearization to provide controllers for nonholonomic robots.

6.1 Two Groups Merging and Continuing to Task Location

The goal in this example (Fig. 8) is to join groups of four and three robots into a single group for a task which requires seven robots. Once the groups are merged and in the desired boundary shape and size, they proceed to the task location. We used as parameters $\delta_{\max} = 2.5$, $\delta_{\min} = 0.2$, $\gamma^{\max} = 5$, $\gamma^{\min} = 0.2$, $\varepsilon = 0.1$, and $s_h^i = s_w^i = 2.5 - 2/N^i$.

6.2 Three Groups Merging and Splitting

In this example (Fig. 9), two tasks require seven and two robots each. Nine robots are available across three groups of three. The three groups merge, then split into

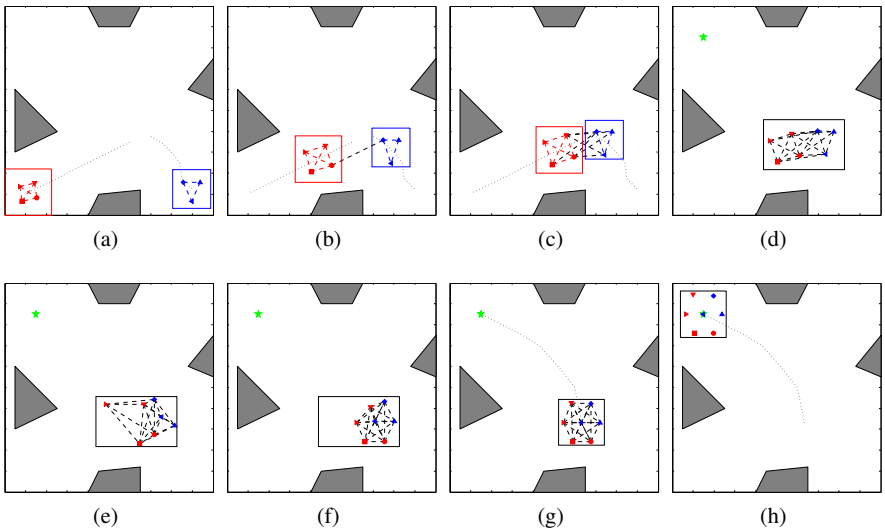


Fig. 8: A two group simulation. Dashed lines represent communication links (omitted in (h)), dotted lines represent the path, and the star represents the task location. (a) Initial condition. (b) Direct inter-group communication established. (c) Merging criterion satisfied. (d) A single group is formed. (e) Mid-reconfiguration. (f) At the desired formation. (g) The box is reshaped. (h) At the task location.

two groups of seven and two robots. The groups now proceed to their respective task locations. Here we used the same parameters as above (except $\gamma^{\min} = 0.5$).

7 Complexity

In this section we discuss the complexity of our method. For each pair of agents with collision constraints we have one annulus with 8 regions, resulting in a maximum

$$P_{max} = 8^{N^i(N^i-1)/2}$$

polytopes in $\mathscr{C}_T^i$. Although this scales exponentially with the number of robots in the group, we only construct the polytopes as we expand nodes in the polytope graph.

To solve Problem 2, we use an A^* algorithm. In an A^* algorithm, the number of nodes expanded is exponential in the actual path length, unless the error of the heuristic grows no faster than the logarithm of the actual cost [16]

$$\left|h\left(\mathscr{F}_d^0, \mathscr{F}_d^g\right) - h^*\left(\mathscr{F}_d^0, \mathscr{F}_d^g\right)\right| \leq O(\log h^*(n)).$$

Although we do not have a bound for our heuristic error, empirically we have found that there exists a path to the goal of the heuristic cost. If there exists a path to the goal, then there likely exist other paths of the same length to the goal (though in the case of differently weighted transitions, equal length may not correspond to equal

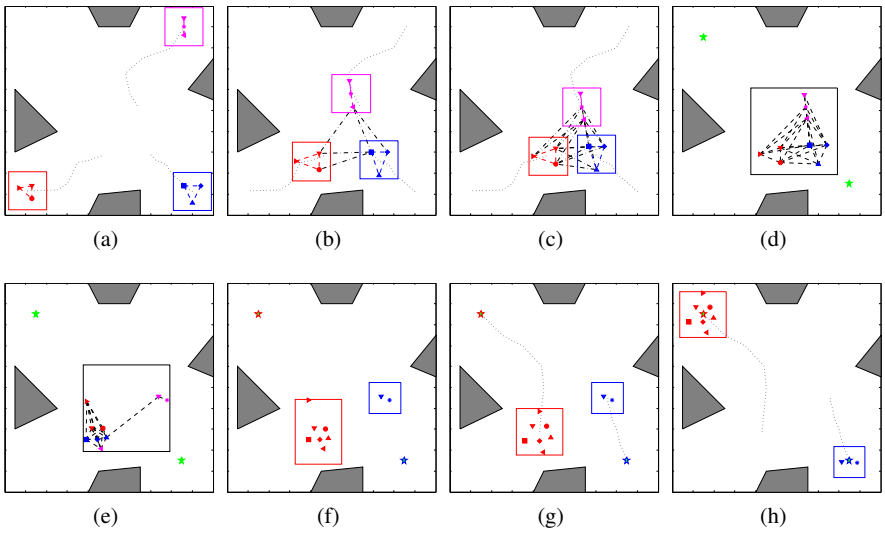


Fig. 9: A three group simulation. Dashed lines represent communication links (omitted (g)-(j) where graphs are complete), dotted lines represent the path, and stars represent task locations. (a) Initial conditions. (b) Inter-group direct communication established. (c) Merging criterion satisfied. (d) One group is formed. (e) Prior to disconnection. (f) Groups split. (g) Boxes are resized. (h) At the task locations.

cost). For example, if the start formation is $\mathscr{F}_d^0 = \{1,2,3\}$ and the goal formation $\mathscr{F}_d^g = \{1,1,1\}$, $h(\mathscr{F}_d^0, \mathscr{F}_d^g) = 3$, there exist three paths (ABE, CBE, CDE in Fig. 5c), with cost $h^*(\mathscr{F}_d^0, \mathscr{F}_d^g) = 3$. In general, since the graph is cyclic, it is likely that a path exists with the exact cost of the heuristic. (If the graph is weighted such that each edge is not of equal cost, this is not generally the case).

We construct a controller in each polytope in $t = \{c_{t^1}^i, \cdots, c_{t^g}^i\}$ to solve Problem 1. The vector field can be computed in $O(\sum_{d=0}^{2N^i} \Phi_d)$ time, where Φ_d is the total number of d-dimensional cells in the GVD [11]. For bounds on Φ_d for a certain class of polytopes, see [8].

The complexity increases linearly in the number of concurrent merging and reconfiguring processes, since each group computes its own controllers.

8 Conclusion

We have presented a method for controlling multiple groups of robots to create, reconfigure, and maintain formations under communication constraints. We provide guarantees of safety, preventing inter-robot collisions and collisions with obstacles in the workspace. Our controller is entirely automatic, and requires information about the space, the desired formation, and the task location. We have discussed briefly the complexity of our approach, which in the worst case scales exponentially in N^i and $h^*\left(\mathscr{F}_d^0, \mathscr{F}_d^g\right)$.

The algorithm is complete based on the choice of abstraction boundary. Since the abstraction is an overestimate of the area occupied by the robots, it is possible that some solutions will be lost. This is especially the case when fixing the boundary of the abstraction while the group is navigating through the space. It should be possible to enforce a minimum area of the space inside the boundary, and maintain discrete formations under some deformations. Further study is required for the case where the abstraction can be reshaped to allow for navigation through cluttered spaces.

Finally, our approach lends itself to planning and control for heterogeneous teams. This is an issue of future research.

References

1. Anderson, C., Franks, N.R.: Teams in animal societies. Behav Ecol **12**(5), 534–540 (2001)
2. Ayanian, N., Kumar, V.: Decentralized feedback controllers for multi-agent teams in environments with obstacles. In: Proc. IEEE Int. Conf. Robot. Autom., pp. 1936–1941. Pasadena, CA (2008)
3. Belta, C., Kumar, V.: Abstraction and control for groups of robots. IEEE Trans. Rob. **20**(5), 865–875 (2004)
4. Desai, J., Ostrowski, J., Kumar, V.: Modeling and control of formations of nonholonomic mobile robots. IEEE Trans. Rob. Autom. **17**(6), 905–908 (2001)
5. Egerstedt, M., Hu, X.: Formation constrained multi-agent control. IEEE Trans. Rob. Autom. **17**(6), 947–951 (2001)
6. Gerkey, B., Vaughan, R.T., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: Proc. of Int. Conf. on Advanced Robotics. Coimbra (2003)
7. Hsieh, M., Loizou, S., Kumar, V.: Stabilization of multiple robots on stable orbits via local sensing. In: Proc. IEEE Int. Conf. Rob. Automat., pp. 2312–2317 (2007)
8. Klee, V.: On the complexity of d-dimensional Voronoi diagrams. Archiv der Mathematik **34**(1), 75–80 (1980)
9. Kvasnica, M., Grieder, P., Baoti, M.: Multi-parametric toolbox (mpt) (2004)
10. Leonard, N., Fiorelli, E.: Virtual leaders, artificial potentials and coordinated control of groups. In: Proc. IEEE Conf. Decis. Contr., vol. 3, pp. 2968–2973 (2001)
11. Lindemann, S.R., LaValle, S.M.: Smoothly blending vector fields for global robot navigation. In: Proc. IEEE Conf. Decis. Contr. Seville, Spain (2005)
12. Michael, N., Kumar, V.: Controlling shapes of ensembles of robots of finite size with nonholonomic constraints. In: Proceedings of Robotics: Science and Systems, pp. 157–162 (2008)
13. Ogren, P., Egerstedt, M., Hu, X.: A control lyapunov function approach to multi-agent coordination. In: Proc. IEEE Conf. Decis. Contr., vol. 2, pp. 1150–1155 vol.2 (2001)
14. Olfati-Saber, R., Murray, R.: Distributed cooperative control of multiple vehicle formations using structural potential functions. In: Proc. of IFAC World Congress. Barcelona (2002)
15. Rimon, E., Koditschek, D.: Exact robot navigation using artificial potential functions. IEEE Trans. Robot. Autom. **8**(5) (1992)
16. Russel, S., Norvig, P.: Artificial Intelligence: a Modern Approach, 1 edn. Prentice Hall, Englewood Cliffs, NJ (1995)
17. Smith, B., Wang, J., Egerstedt, M.: Persistent formation control of multi-robot networks. In: Proceedings of the IEEE Conference on Decision and Control, pp. 471–476 (2008)
18. Tanner, H.G., Jadbabaie, A., Pappas, G.J.: Flocking in fixed and switching networks. IEEE Trans. Automat. Contr. **52**(5), 863–868 (2007)
19. Yang, P., Freeman, R., Lynch, K.: Multi-agent coordination by decentralized estimation and control. IEEE Trans. Automat. Contr. **53**(11), 2480–2496 (2008)