2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Coalition Coordination for Tightly Coupled Multirobot Tasks with Sensor Constraints

Yu Zhang and Lynne E. Parker and Subbarao Kambhampati

Abstract— **Although many approaches have been developed to form robot coalitions that can achieve a multirobot task, no general methods exist to execute these coalitions, especially when the coordination among the robots is tightly coupled. In this paper, we propose a coordination mechanism as the first step to address coalition execution; it provides a flexible method to reason about synergies with overlapping coalitions (thus enabling multi-tasking robots in multi-robot tasks), which not only improves efficiency, but also reduces resource requirements in task execution. This means that our approach enables tasks that cannot be easily handled before, especially when critical resources are rare but commonly required. Our approach is based on the concept of sensor constraint, which is introduced by the tight coupling (e.g., information sharing) between the robots. We show that our algorithm is sound and complete in finding a coordination solution given a few assumptions, and discuss a distributed implementation. Simulation results are provided to demonstrate the capabilities of this new approach.**

I. INTRODUCTION

To accomplish a single multirobot task,[1] robots must execute the coalitions that are formed for the task. However, general methods for coalition execution do not yet exist, especially for tightly coupled multirobot tasks, in which close coordination with information sharing between the robots is required. This coordination must be determined dynamically in the current situation. For example, in a cooperative robot surveillance and monitoring task, various types of robots (including smart sensors) are distributed in the workspace. The goal is to monitor the environment and report the locations of any unusual objects. If only a handful of robots can localize, the team needs to coordinate tightly with each other via information sharing. In this scenario, a robot that can localize may potentially need to assist multiple robots (or coalitions) at the same time; to ensure coverage, the mobile robots also need to move around in the environment. We are immediately faced with a challenging coordination scenario.

To address the coordination problem in a general manner, first, a method is needed to model the interactions among robots in the coalitions and the environment; these interactions are dynamically determined in the current situation. With this modeling, we show that the coordination problem of coalition execution can be transformed to the maintenance of the required sensor constraints for the task, which specify configuration constraints on the coalition members. As an example, consider a cooperative robot navigation task, in which one robot has a localization capability and one does not. Both robots have a fiducial sensor to detect teammates ahead. There exists a sensor constraint that is introduced by the fiducial information, which requires one robot to keep the other in its sensor field of view (FOV). Another example is a robot swarming task that requires close proximity, in which the retrieval of the bumper information requires the adjacent robots to keep in contact with each other. Based on the set of sensor constraints, our approach allows robot resources to be shared among coalitions, thus enabling tasks that cannot be easily achieved before, when certain resources are rare.

After a brief discussion of the related work in Section II, we present the new coordination mechanism in Section III with discussions of a distributed implementation. Finally, results are presented in Section IV following by conclusions.

II. RELATED WORK

While many approaches have been provided to form coalitions [3], [4], [9], [13], [15], [16], coalition execution is often assumed to be handled by the preprogrammed behaviors on individual robots. Although this may be true for multirobot tasks that are loosely coupled, in which the task can be divided into subtasks that can be accomplished by individual robots, it is not so much for tightly coupled tasks. These tasks require robots to interact in a closely coordinated manner, which must often be dynamically determined. Although application-specific methods [7], [14] can be applied, they do not generalize or scale.

To create a general solution, first, interactions among robots and the environment must be specifically modeled. Techniques to form coalitions are introduced to reason about the groups of robots necessary to accomplish the given task. The formation process of coalitions, in turn, determines the required interactions. However, due to the dynamism of these interactions in tightly coupled multirobot tasks, approaches [3], [4], [15] that only address loosely coupled tasks or reason about forming coalitions based statically on robot capabilities do not suffice. Among the approaches that are capable of dynamic modeling, the approaches that are built on the concept of information [13], [16] have been shown to provide more flexibility with autonomous information sharing. Hence, we adopt this formulation in this work.

In order to execute coalitions to accomplish the task, the robots must coordinate their behaviors based on the set

This research is supported in part by the ARO grant W911NF-13-1-0023, and the ONR grants N00014-13-1-0176 and N00014-13-1-0519.

Yu Zhang and Subbarao Kambhampati are with the Yochan research group at the Arizona State University, Tempe, AZ 85281, USA, {yzhan442,rao}@asu.edu

Lynne E. Parker is with the Distributed Intelligence Laboratory at the University of Tennessee, Knoxville, TN 37996, USA, {parker}@eecs.utk.edu

[1] In this paper, we consider coalition execution for an individual task; task allocation (e.g., [3]) and scheduling (e.g., [18]) are not considered.

of sensor constraints introduced by the interactions. One coordination approach is to employ planning techniques [1], [12] to plan execution paths that satisfy the requirements of these constraints. The planned paths can then be used to feed back commands to control the robots. However, this approach is computationally expensive and hence not scalable. A more suitable approach is to use distributed control methods [2], [5], [10] to maintain the robot configurations required for the constraints locally. Although the coordinated robot behaviors in this work may appear similar to these control methods, there is a fundamental difference. While the set of configuration (sensor) constraints are given in these control methods, in this paper, it is determined automatically and dynamically based on the information requirement. This automatic process also provides a flexible method to reason about synergies with overlapping coalitions. For example, if two robots without a localization capability share the goal paths (i.e., one path aligns with the other), they may rely on the same robot and navigate together. While task synergy [8], [11] for single task robots [6] has been studied before, to the best of our knowledge, this is the first work that investigates synergies of coalitions in a single multirobot task, thus enabling multi-tasking robots in multi-robot tasks.

III. COALITION COORDINATION

In this section, we first provide a brief introduction of the approach to model the interactions based on the concept of information. Based on this modeling, coalitions are formed and sensor constraints are introduced. We implement our approach on one of the recent architectures that use this approach [17], which extends [16] and provides an essential capability to execute coalitions: forming executable coalitions. While using a specific architecture, our discussion emphasizes its general components that need to be present in formulating multirobot tasks to enable a similar modeling capability. Then, we discuss the new coordination mechanism with a single coalition in Section III-C and extend it to multiple coalitions in Section III-D.

A. IQ-ASyMTRe

Sections III-A and III-B serve primarily as background knowledge for the IQ-ASyMTRe architecture [17], which is used in this paper to model the interactions and form coalitions. Based on schema theory, in IQ-ASyMTRe, a robot is composed of sensors (ESs), communication schemas (CSs), computation or perceptual schemas (PSs), and motor schemas (MSs). Schemas accept input information and can optionally output information; schemas can be activated when their input information is provided. The goal of forming a coalition is to connect these schemas to activate a desired MS for the task (referred to as a *task MS* henceforth). Figure 1 provides an example of the potential local schema connections for robot R_1 to activate a navigation MS, with each schema being represented as a rectangle box except for the *tOR*. In Figure 1, EPS and RPS are special types of PS. An RPS is introduced to perform information conversions. For a few commonly used examples, see Table I; an EPS is always associated with an ES to process raw sensor data. We will refer back to this example.

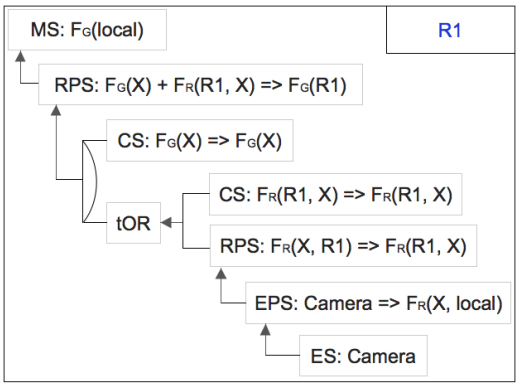


Fig. 1. IQ-ASyMTRe [17]: solution space (as an *and-or* tree) for a robot to obtain its global position for a navigation MS with only a camera sensor. The referent *local* refers to the robot itself. The solution space encodes two potential solutions. One solution is to have another robot send over its global position (CS: $F_G(X) \Rightarrow F_G(X)$) and use the camera to sense the relative position of that robot (EPS: Camera $\Rightarrow$ $F_R(X, local)$). An RPS (RPS: $F_R(Y, X) \Rightarrow F_R(X, Y)$) is used to convert $F_R(X, R_1)$ to $F_R(R_1, X)$. The other solution (*tOR*) is to have both information instances (CS: $F_G(X) \Rightarrow F_G(X)$ and CS: $F_R(R_1, X) \Rightarrow F_R(R_1, X)$) sent over by another robot.

TABLE I
IQ-ASyMTRe [17]: EXAMPLES OF RPS'S

RPS	Description
$F_G(X) + F_R(Y, X) \Rightarrow F_G(Y)$	global + relative $\Rightarrow$ global
$F_R(Y, X) \Rightarrow F_R(X, Y)$	relative $\Rightarrow$ relative
$F_R(X, Z) + F_R(Y, Z) \Rightarrow F_R(X, Y)$	relative + relative $\Rightarrow$ relative

To form a coalition for a robot that is assigned an MS (referred to as a *task robot* henceforth), IQ-ASyMTRe reasons based on these local connections, which allow information to flow between different schemas within the robot itself or from other robots through CSs, in order to satisfy the input of the MS. All robots necessary to activate the MS are considered to be in the coalition.

To ensure valid schema connections, the output schema that provides information must be able to produce the information that the input schema accepts. Furthermore, when forming a coalition, the actual information must be validated to ensure a match. To reason about this, IQ-ASyMTRe uses the concepts of *information type* and *information instance* to label the input and output of schemas, which are used to specify the semantic meaning of information. For example, while *robot's position* conveys the semantic meaning, *physical coordinates* is the value associated with the meaning. The semantic meaning of information specifies what the information describes (e.g., position), and the referents to which it applies (e.g., the robot).

Definition 3.1 (Information Type): Given an application domain, an information type (denoted by $\mathcal{F}$) is a unit to specify non-referent related information semantics. □

Information types for different domains are often different. In general, one should minimize the number of information

types in a given domain in order to reduce the search space.

Definition 3.2 (Information Instance): An information instance, or $F(\mathcal{E})$ (F for short), is an information type with the associated referent set $\mathcal{E}$ ($\mathcal{E}$ is an ordered set). The number of referents or $|\mathcal{E}|$ is determined by the type. □

For example, in Figure 1, F_G and F_R refer to the global and relative position information, respectively. $F_G(X)$ is associated with one referent X. Each referent may be statically instantiated to an entity, which can be any identifiable object in the environment, or remain uninstantiated for future or dynamic instantiations. Single uppercase letters are used in this paper to denote uninstantiated referents. IQ-ASyMTRe also requires the uninstantiated referents of the same label within the same context (e.g., in a schema) to be instantiated to the same entity (referred to as *referent instantiation constraint*), and that the referents of the same information instance be instantiated to different entities.

We return now to the validation of schema connections. While this validation must be performed based on information instances that are not fully instantiated, the process to form a coalition must ensure that the information instances, after full instantiations according to the actual information, satisfy the referent instantiation constraints. Note that dynamic instantiations must be performed via sensing.

B. Coalition Solution

A coalition includes all the robots in the coalition solution. To create the coalition solution, IQ-ASyMTRe first connects local schemas to create the solution space as shown in Figure 1. This process always starts with the input of the MS (i.e., the information sink node), which is then connected to the outputs of schema nodes that can provide the required information instances. These upstream nodes are then connected in a recursive manner (for more details, see [17]), until reaching the information sources, which are CSs or ESs. Note that information sources can be dummy ES nodes, which are created for information that remains constant and is known a priori. For example, an environment map.

After the solution space is created, the next step is to search for coalition solutions. Note that while the solution space encodes *potential solutions* to activate the MS using local schema connections, the coalition solution is an instantiation of one of the potential solutions (i.e., making a choice at each *tOR* node) with a complete specification of external interactions. Figure 2 provides an example of a coalition solution for one of the potential solutions in Figure 1. Comparing Figure 1 with Figure 2, more specifically, information instances in solution spaces may not be fully instantiated (e.g., $F_G(X)$ in Figure 1) but they are in coalition solutions; a solution space can include multiple potential solutions when there are *tOR* nodes to provide schema connection options; a solution space pertains only to one robot while a coalition solution can span multiple robots; the leaf nodes, or information sources, in coalition solutions are always ESs.

With the solution space, IQ-ASyMTRe selects a coalition solution and sets up the initial coalition as follows:

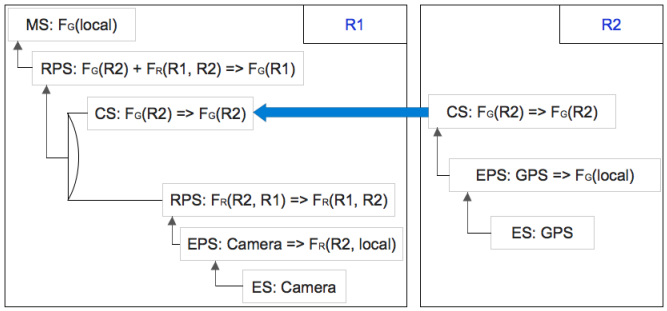


Fig. 2. IQ-ASyMTRe [17]: A possible coalition solution for one of the potential solutions in Figure 1.

- Check each potential solution with potential coalition members to instantiate the solution in order to validate the coalition solution;
- Select a valid coalition solution and set up the coalition.

C. Coalition Coordination

We can now start the discussion of coalition coordination. In this section, we start with a single coalition. One observation is that the only precondition for the associated task MS to keep in activation is the constant provision of the input information during execution, which can be satisfied as long as the coalition solution is in effect. This directly translates to the maintenance of the coalition solution.

Similar to [17], assuming that the robots are always within each other's communication range[2], the only factors that can influence the coalition solution are the leaf nodes (i.e., ESs). In order to provide the information to the downstream nodes, each ES may introduce a configuration constraint on the related entities. We refer to this configuration as *information configuration* and the constraint as *sensor constraint*.

Definition 3.3 (Information Configuration): The configuration of an information instance $F(\mathcal{E})$, denoted by $cf(F(\mathcal{E}))$, represents the set of all semantic configurations (for $\mathcal{E}$) that are included in the specification of $F(\mathcal{E})$. □

Semantic configuration is a general term here, which can specify, for example, physical configuration and state. Consider $F_R(X,Y)$ as an example. $cf(F_R(X,Y))$ specifies a relationship of relative physical placement between X and Y. $cf(F_R(X,Y))$ includes, for example, the same relationship between X and Y along each coordinate axis, and that X and Y are physically located in the workspace. Information configuration encodes the semantic meaning expressed by the information without the value; it is a general concept for information since it is associated with all information representations (e.g., information instance in this paper).

Definition 3.4 (Sensor Constraint): A sensor constraint is a special type of information configuration constraint (i.e., a restriction imposed on $cf(F(\mathcal{E}))$), in which $F(\mathcal{E})$ is produced by the sensor. A sensor constraint has the form of $e \rightarrow_{F(\mathcal{E})} E$, in which $e \in \mathcal{E}$ and $E = \mathcal{E} - e$. □

[2]Although neither [17] or the approach in this paper relies on this assumption to work, it simplifies the discussions. In most cases, communication divides the team into disjoint groups, which can then be separately considered.

Note that e always refers to the robot equipped with the sensor (or the local robot). For example, the relative position information from the camera sensor in Figure 1 introduces a sensor constraint $local \rightarrow_{F_R(X,local)} \{X\}$, which is instantiated to $R_1 \rightarrow_{F_R(R_2,R_1)} \{R_2\}$ in Figure 2.

To execute a coalition, the coalition members need to maintain the set of sensor constraints in a coordinated manner while the task robot of the coalition executes the task MS. Using IQ-ASyMTRe to form individual coalitions ensures that the coalition is initially executable. Hence, all the sensor constraints are initially satisfied as well. However, to accomplish the task, we also need to determine whether these sensor constraints can be maintained during execution. For example, in the navigation task, a task robot without a localization capability needs to identify that a coalition with a robot that is immobile is not a solution, even though this other robot can localize; neither is the case when the other robot is mobile but executing a path to move in the opposite direction. Next, we discuss how robots can reason about this in various situations and maintain the constraints whenever possible. To achieve this, we need to first understand the relationship between sensor constraint and individual robot configurations.

Given the related information types of the domain, the configuration of individual robots can also be specified using information configurations. The multi-referent configuration that is associated with the sensor constraint is encoded in the configurations of individual referents. For example, in the navigation task, $cf(F_R(R_2, R_1))$ can be encoded jointly by $cf(F_G(R_1))$ and $cf(F_G(R_2))$. This reasoning is directly related to RPSs. Since it involves multiple information instances, we first introduce the following definition:

Definition 3.5 (Information Instance Set): An information instance set (IIS) is an unordered set of information instances. □

We then associate RPSs with the notion of *information inference*.

Definition 3.6 (Information Inference): Given an IIS s and an information instance F, s can infer F if $F \in s$, or F can be converted by RPSs using information instances that are in s or that are inferred from s. □

For example, $F_R(X, local)$ can be inferred from $\{F_G(X), F_G(local)\}$ using the second RPS in Table I. Also, note the recursive definition. We further introduce *power set*.

Definition 3.7 (Power Set): The power set of an IIS s, denoted by $P(s)$, is defined as

$$P(s) = \{F : s \text{ infers } F\}$$

□

To collectively specify the configuration of multiple referents, we define *Joint Information Configuration*.

Definition 3.8 (Joint Information Configuration): Given an IIS s, its joint information configuration, denoted by $cf(s)$, is defined as:

$$cf(s) = \bigcup \{cf(F) : F \in P(s)\}$$

□

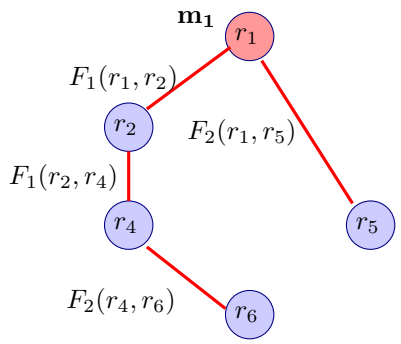


Fig. 3. Example of a constraint graph for a single coalition.

Given the above discussions, we can conclude that $cf(\{F_G(R_1), F_G(R_2)\}) \supseteq cf(F_R(R_2, R_1))$. Given a sensor constraint on $cf(F_R(R_2, R_1))$, this proposition implies that we can maintain it by giving freedom to all but one of the individual configurations on its left hand side,[3] which is useful for coalition execution since the task robot must execute the task MS while others can change their individual configurations to maintain the constraint. Given a sensor constraint $e \rightarrow_{F(\mathcal{E})} E$, the idea is to find a set of individual configurations $\{cf(F(e))\}_{e \in \mathcal{E}}$ such that $cf(\{F(e)\}_{e \in \mathcal{E}}) \supseteq cf(F(\mathcal{E}))$. When there are multiple options to break $cf(F(\mathcal{E}))$ in such a way, each option can be considered independently.

During execution, if an robot e has an MS that can update its configuration $cf(F(e))$ according to $cf(F(\mathcal{E}))$, we refer to the robot as being *compatible* with the sensor constraint and the MS is referred to as a *constraint MS* for $F(\mathcal{E})$. In the simplest case, if e is not the task robot and no other MS is assigned to it, this MS can be activated on e to maintain the sensor constraint. If all referents except one in $\mathcal{E}$ are compatible, the constraint becomes *satisfiable*; one way to satisfy it then is to activate the constraint MSs on these referents. This reasoning requires the following definition:

Definition 3.9 (MS Type): Given the input IIS s, the type of MS, denoted by $\mathcal{T}$(MS(s)), is an information instance F such that its value is updated by the MS when activated.[4] □

For example, given the MS in Figure 2, denoted here by MS_{nav}, since we know that navigation updates the robot position, we have $\mathcal{T}(\text{MS}_{nav}(\{F_G(local)\})) = F_G(local)$. Note the referent instantiation constraint here; by executing the MS, a robot also instantiates the MS type, e.g., $\mathcal{T}(\text{MS}_{nav}(\{F_G(R_1)\})) = F_G(R_1)$. Even when an entity e is not a robot, we may still be able to specify its motion type. In this case, we use a similar notation and directly write, e.g., $\mathcal{T}(e) = F_G(e)$. When a robot is r is executing no more than one MS, $\mathcal{T}(r)$ can also be conveniently used to refer to the type of the currently active MS on r.

Now, we are ready to discuss coalition coordination with

[3]One underlying assumption is that the RPS must remain valid when exchanging the information instance on the right hand side with any instance on the left. This is true for all the RPSs that we use.

[4]To simplify the discussion, if an MS can influence the configurations of more than one information instance, it is considered as multiple MSs.

TABLE II

RPS'S FOR FIGURES 3 AND 4

RPS
$F_4(X) + F_4(Y) \Rightarrow F_1(X, Y)$
$F_4(X) + F_4(Y) \Rightarrow F_2(X, Y)$
$F_4(X) + F_4(Y) + F_4(Z) \Rightarrow F_3(X, Y, Z)$

a single coalition. First, we define *constraint graph*.

Definition 3.10 (Constraint Graph): The constraint graph is an undirected graph, in which each node represents an entity; two nodes are connected if both are present in the same sensor constraint. Each edge is labeled by the information instance associated with the constraint. □

When two nodes are present together in more than one constraint, they are connected by multiple edges, thus forming loops. Discussion of this situation is delayed until Section III-D. Figure 3 provides an example of a constraint graph with a single coalition (i.e., with a single task MS m_1), in which r_1 is the robot that is assigned m_1. Other robots provide information to help r_1 activate m_1. While a sensor constraint is directed, the associated edges in the graph are often not, since they specify configuration constraints. (Directed edges only introduce asymmetry on the process to check coordination solutions when present.)

Proposition 3.1: Given a constraint graph for a single coalition that has only robot entities (controllable), if all entities are compatible with their respective sensor constraints, the coalition can satisfy the set of constraints if the graph has a tree or forest structure. □

Proof: (Sketch) Starting from the node with the task MS, for each node v that connects to it, we can activate the constraint MS on v to maintain the respective sensor constraint. Note that there is no need to check the node that we have already checked. Continue this process recursively. Since the tree structure ensures that we do not come back to the previous nodes (which already have an assigned MS), this process would assign no more than one MS to any robot. This conclusion clearly holds for other trees in a forest. ■

For the scenario in Figure 3, suppose that we have the RPSs in Table II and the RPSs to switch the referent ordering for F_1, F_2 and F_3 (similar to the second RPS in Table I), $\mathcal{T}(m_1\{F_4(local)\}) = F_4(local)$, and that all robots can activate the constraint MS m_c^1 and m_c^2, which satisfy $\mathcal{T}(m_c^1(\{F_1(X, local)\})) = F_4(local)$ and $\mathcal{T}(m_c^2(\{F_2(X, local)\})) = F_4(local)$. Based on Proposition 3.1, we can conclude that a coordination solution exists.

D. Coordination with Multiple Coalitions

When there are loops, multiple task MSs or non-robot entities (uncontrollable) in the constraint graph, coalition coordination becomes more complex. In this section, we show how these situations can be incorporated. Our approach also allows different coalitions to share robots.

A more complex scenario is presented in Figure 4. First, we realize that certain domain knowledge is unavoidable and desirable. For example, in a box pushing task, although

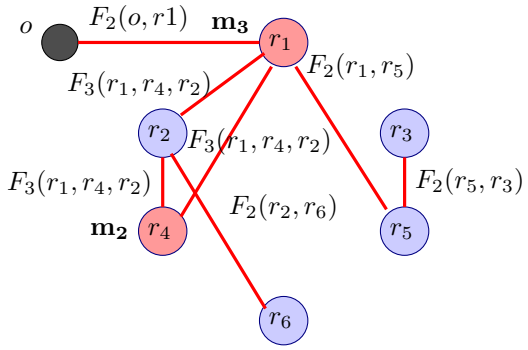


Fig. 4. Example of constraint graph for a multirobot task that includes two task MSs (m_2 and m_3), a loop and a non-robot entity o.

there is a sensor constraint introduced by the requirement of $F_R(box, local)$, it does not influence the execution since the box always moves along with the robot. The domain knowledge is captured by the concepts of *domain compatibility* and *disjoint configuration*, which are assumed to be specified either a priori in the task specification or dynamically by a higher level planner.

Definition 3.11 (Domain Compatibility): Given two task MSs, domain compatibility specifies the configuration constraint (in terms of an information instance) that is satisfied by the task robots when executing these MSs. □

In the box pushing task, the domain compatibility can be specified as $DOM(m(box), \text{MS}_{push}(\{F_R(box, local), ...\})) = F_R(box, local)$, in which $m(box)$ represents the motion (a virtual MS) of the box and $\text{MS}_{push}(\{F_R(box, local), ...\})$ represents the box pushing MS (the input is only partially specified for brevity). Domain compatibility is useful when both entities are already assigned an MS and there is a sensor constraint between them. In this case, if these two MSs are domain compatible with respect to the constraint, we do not need to activate a constraint MS.

Definition 3.12 (Disjoint Configuration): Given two information instances F_1 and F_2, they are of disjoint configuration if $cf(F_1) \cap cf(F_2) \equiv \emptyset$. □

Given a robot e that is involved in a constraint, even if e is already assigned an MS m', if $cf(\mathcal{T}(m')) \cap cf(\mathcal{T}(m)) \equiv \emptyset$, in which m is the constraint MS, we can activate m on e. In this case, m' and m can be considered as orthogonal components of a single MS. For example, if r_1 and r_2 are both assigned an MS to explore the workspace and there is a constraint that requires them to keep their fiducial sensors at the same height (in order to detect each other), it is most likely that the MS to change the sensor's height and the exploratory MS do not influence each other. However, note that $cf(F_1) \cap cf(F_2) \equiv \emptyset$ is a strong condition. In practice, domain compatibility often provides more flexibility.

Definition 3.13 (Propagated Configuration Constraint): Given an IIS s that is associated with the currently satisfied configuration constraints (as a result of domain knowledge or the activations of constraint MSs), a propagated configuration constraint is defined as a constraint that is

associated with any instance in $P(s)$. □

Configuration propagation is used to automatically infer *implied* constraints given the currently satisfied constraints. For example, given that the constraints associated with both $F_R(r_1, r_2)$ and $F_R(r_1, r_3)$ are initially satisfied and maintained, from the third RPS in Table I, we know that a constraint on $cf(F_R(r_2, r_3))$ is also maintained. Hence, in such a case, even if both r_2 and r_3 are already assigned an MS, we can ignore the influence of this sensor constraint if it is initially satisfied. Note that in this case, configuration propagation specifies a transitive relationship.

Again, assuming that the robots have the necessary constraint MSs and the RPSs in Table II, Table III presents a set of conditions for Figure 4; adding the constraint MS for F_3 to the robots then ensures a solution. The coordination algorithm for multiple coalitions is presented in Algorithm 1. The algorithm first pushes all nodes (i.e., robots) onto a queue and invokes *Coordinate* recursively, which checks each edge to determine a coordination solution.

Algorithm 1 Algorithm for Coalition Coordination

1: **INPUT**: a constraint graph $G = (V, E)$; domain compatibilities D; disjoint configurations C.
2: Create a queue Q and push in all nodes (i.e., robots).
3: Create a set s for currently satisfied constraints.
4: Create a set c for checked nodes and edges.
5: Add domain compatibilities to s.
6: $node$ = POP(Q); Coordinate($node, G, C, D, Q, s, c$).
7:
8: **PROCEDURE**: Coordinate(v, G, C, D, Q, s, c):
9: **while** $(U = \{u : (u, v)_F \in E \ \&\ (u, v)_F \notin c\}) \equiv \emptyset$ **do**
10: If $Q \equiv \emptyset$, return $true$; else $c = c \cup v$, v = POP(Q).
11: **end while**
12: Add $(u, v)_F \in U$ to c.
13: **if** c_1: $cf(F) \not\subseteq cf(s)$ **then**
14: **for all** $x \in \{u, v\}$ **do**
15: **for all** constraint MS m for F on x **do**
16: **if** c_2: $cf(\mathcal{T}(x)) \cap cf(\mathcal{T}(m)) \equiv \emptyset$ or $cf(\mathcal{T}(x)) \equiv \emptyset$ **then**
17: G_x^m = Duplicate(G); assign m to x in G_x^m.
18: s_x^m = Duplicate(s); add F to s_x^m if the constraint is satisfied.
19: Add domain compatibilities to s_x^m.
20: r_x^m = Coordinate($v, G_x^m, C, D, Q, s_x^m, c$).
21: **else**
22: $r_x^m = false$ (no solution for current x and m).
23: **end if**
24: **end for**
25: **end for**
26: **return** $\vee_{x,m} r_x^m$.
27: **end if**
28: **return** Coordinate(v, G, C, D, Q, s, c).

Lemma 3.2: Algorithm 1 is sound: the constructed coordination solution is correct. □

Proof: (Sketch) For any node, Algorithm 1 only assigns it an MS m if one of the following conditions holds: 1) If the node can execute m and that it is not already assigned an MS; 2) m is orthogonal to the assigned MS (line 16). Given that a constraint between any two nodes u and v (every edge is checked once) is initially satisfied, the set of constraints is maintained as a result of either domain capability, configuration propagation or the activations of constraint MSs; otherwise, the algorithm returns no solution. ■

TABLE III
ADDITIONAL CONDITIONS FOR FIGURE 4

1.	$\mathcal{T}(m_2\{F_4(local)\}) = F_4(local)$
2.	$\mathcal{T}(m_3\{F_2(o, local)\}) = F_4(local)$
3.	$DOM(m(o), m_3(\{F_2(o, local)\})) = F_2(o, local)$
4.	$cf(F_4) \cap cf(F_3) \equiv \emptyset$

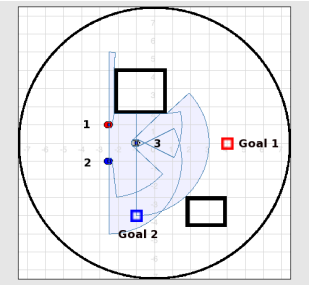
Lemma 3.3: Given a constraint graph, Algorithm 1 is complete in finding a coordination solution, when given only the RPSs, domain compatibilities and disjoint configurations of the domain. □

Proof: (Sketch) To satisfy a sensor constraint, either we need to activate a constraint MS, or the constraint is already satisfied as a result of domain compatibilities or configuration propagation. Since Algorithm 1 checks all such possibilities for a coordination solution (i.e., returns $\vee_{x,m} r_x^m$), and it ensures that no MSs are introduced unnecessarily (through c_1 and c_2), the search is complete. ■

The complexity is dominated by the computation of $P(s)$ for $cf(s)$, and is expensive in general (i.e., exponential in the numbers of information types and RPSs). However, depending on the problem domain, this complexity can be reduced. For example, when the configuration propagation is transitive, the complexity becomes linear.

E. Distributed Implementation

In a distributed implementation, each task robot starts with a constraint graph with only itself and no edges. After a task robot finds a set of coalition solutions, it ranks these solutions based on their costs (see [17] for details). For the current solution being considered, this task robot updates its local graph accordingly and runs Algorithm 1 to compute a coordination solution for the local coalition. If a solution is found, this task robot shares the set of newly assigned MSs along with the set of associated sensor constraints with the other task robots, in order for them to update the graph accordingly. At any time, only one robot is allowed to perform such an update. In case that no solution is found, the robot can backtrack these updates and request the robot with the most recent update to change its coalition solution. The combined coordination solution of all local coalition solutions becomes the final coordination solution. Although this process can be expensive, the fact that the local coalition solutions are ranked often implies that the combination that is most likely to produce a final solution is considered first. RPSs, domain compatibilities and disjoint configurations are assumed to be common knowledge in the process.



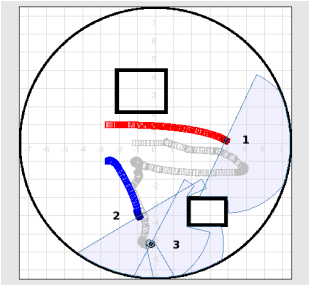
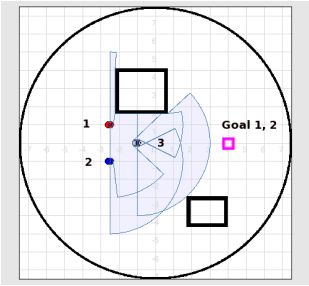


Fig. 5. Scenario 1 in the robot navigation task. The left figure shows the initial robot configurations with goal markers and the right figure shows the final robot configurations with execution traces. The ranges of the laser (fiducial) sensors are also shown.



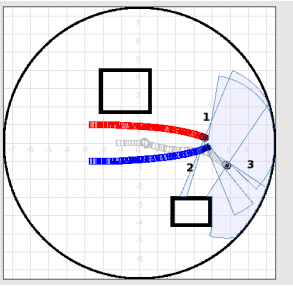


Fig. 6. Scenario 2 in which robots share the same goal location.

IV. SIMULATION RESULTS

Due to many constraints, in this section, we concentrate on only one task domain: cooperative robot navigation; we show that even with very simple scenarios in this simple domain, solutions are not always obvious. Our approach is applied to these scenarios to demonstrate how they are handled in a flexible way, such that it can potentially enable tasks that cannot be easily achieved before, especially when certain resources are limited. The application to more complex tasks will be future work. Simulations are run on a 2.4GHz laptop with 2GB memory, using Player and Stage. For all scenarios, the task includes two navigation MSs, which are to be activated on robot 1 and 2 (without a localization capability), respectively. All robots are equipped with a fiducial sensor to detect nearby teammates. RPSs used are in Table I.

1) Scenario 1: Figure 5 shows the most common scenario in which robots are assigned different goal locations. Both robot 1 and 2 try to set up a coalition with the only robot (labeled 3) that can localize. In this case, each introduces a sensor constraint in the form of $local \rightarrow_{F_R(3,local)} \{local\}$, in which $local$ represents either robot 1 or 2. Robot 1 first sets up the coalition, and informs 2 that it requires a constraint MS to be activated on 3 in order to maintain its constraint. When robot 2 receives this information, since it also requires a constraint MS to be activated, it realizes that this MS would conflict with the MS already assigned to 3, based on the RPSs in Table I. As a result, no coordination solution exists since robot 1 has only one local solution. Hence, the task specification must be updated. In this case, we manually divide (for now) the original task into three tasks. Robot 1 sets up a coalition to execute the first task and the configuration constraint (i.e., relative positioning) is maintained by the constraint MS on robot 3, given the information $F_R(3,1)$. Once the first task is accomplished, robot 3 returns to 2 (the second task) and, through a similar process, helps robot 2 reach the goal (the third task).

2) Scenario 2: The only difference in the second scenario (Figure 6) from scenario 1 is that the robots share the same goal location. Given this scenario, we can add a new condition, $DOM(m(1), m(2)) = F_2(2,1)$, based on the planned paths. Everything remains the same as in scenario 1, before robot 2 receives robot 1's update. In this case, first, robot 2 knows that the constraint on $F_R(3,1)$ is satisfied from robot 1's update. Based on the domain compatibility and the third RPS in Table I, it realizes that the constraint on $F_R(2,3)$ is also satisfied from configuration propagation. Hence, it does not need to activate the constraint MS nor does it require robot 3 to do so. In this case, we have a coordination solution and hence the task can be executed. The configuration constraint associated with $F_R(3,1)$ is maintained by the constraint MS on 3, while the constraint associated with $F_R(2,1)$ is implicitly satisfied, given that robot 2 moves along with 1.

Depending on the scenario, the coordination solution can be very different. For example, even though the two robots share the same goal location, the planned paths may be vastly different due to the environment settings or the path planning algorithm; on the other hand, two robots can still satisfy $DOM(m(1), m(2)) = F_2(2,1)$ even when they do not share the same goal location, e.g., one robot's goal location lies on the path of the other. All these cases can be captured similarly in our approach, thus enabling robot resources to be easily shared whenever possible.

3) Scenario 3: One may argue that when domain compatibility and disjoint configuration information are provided, the proposed reasoning process is unnecessary. Our reply is: Domain compatibility and disjoint configuration only capture domain dependent information and thus are not sufficient to determine a coordination strategy. For example, the domain compatibility in scenario 2 only indicates the closeness of the planned paths for the task robots; it does not specify how other helper robots should behave and whether or not they incur conflicts in the task execution.

Another argument is that why not simply hardcode in the coordination solutions. For example, given the domain compatibility in Scenario 2, one can state that robot 1 and 2 can share a helper robot. Unfortunately, such an approach becomes impractical when the problem becomes more complex. In fact, we show here that the coordination solutions may not be obvious even in simple scenarios, such that it is prone to miss valid solutions or introduce invalid solutions. In this scenario (Figure 7), the robots start in different initial configurations. The coordination solution is created similarly in two phases: Robot 2 first sets up a coalition with 3 and

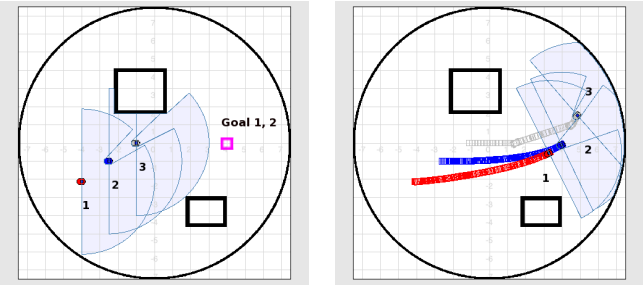


Fig. 7. Scenario 3 with different robot initial configurations.

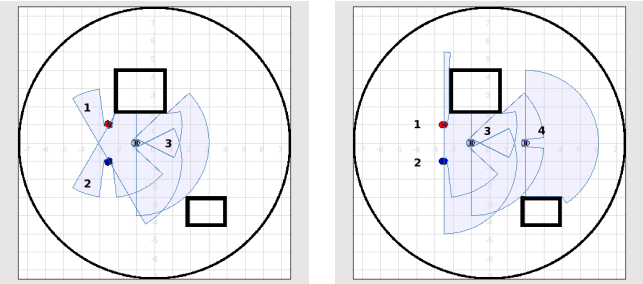


Fig. 8. Scenarios 4 and 5 with robots that are immobile.

updates with 1; Robot 1 then realizes that 2 can localize with the help from 3 so it sets up a coalition with 2, except in this case that 3 is the hidden helper and hence also included in the coalition. Robot 1 then requires two sensor constraints to be satisfied for $F_R(3,2)$ and $F_R(2,1)$. Since $F_R(3,2)$ is already satisfied by robot 2's coalition, it can be ignored; $F_R(2,1)$ is directly supplied by the domain compatibility. Note that in this case, the two robots do not share the same helper robot but a coordination solution still exists.

4) Scenarios 4 and 5: One may then state that the two robots can share a helper robot or help each other. In scenario 4 (the left figure in Figure 8), we let robots 1 and 2 face each other and remove the mobility from 3. In this case, although both robots can localize with the help from 3, which means that they can 'help' each other, no coordination solution exists. How about further enforcing the helper robots to be mobile? In the last scenario, we have a mobile robot 3 that cannot localize but a immobile robot 4 that can. Again, no solution exists. One may ultimately state that the helpers, and hidden helpers, or essentially, all coalitions members must be mobile. This may be a sufficient solution, but unnecessary in some cases, e.g., with an overhead camera (we can add a domain compatibility for $F_R(X, camera)$ in our approach).

V. CONCLUSIONS

This paper presents a coalition coordination mechanism for tightly coupled multirobot tasks, which represents the first step toward a general approach for coalition execution. We use an existing approach for forming coalitions to model the interactions among the robots and the environment. These interactions introduce a set of sensor constraints that must be maintained. Our approach can automatically search for a coordination solution given this information. We show that this approach is sound and complete given a few assumptions, and provide discussions on a distributed implementation. Moreover, this new coordination mechanism provides a flexible method to reason about synergies with overlapping coalitions, thus enabling multi-tasking robots in multi-robot tasks. To the best of our knowledge, this is the first work in the above aspects.

REFERENCES

[1] N. Ayanian and V. Kumar. Decentralized feedback controllers for multiagent teams in environments with obstacles. *IEEE Transactions on Robotics*, 26(5):878–887, Oct. 2010.

[2] L.E. Barnes, M.A. Fields, and K.P. Valavanis. Swarm formation control utilizing elliptical surfaces and limiting functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(6):1434–1445, Dec. 2009.

[3] S.C. Botelho and R. Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2:1234–1239, 1999.

[4] M.B. Dias and A Stentz. A free market architecture for distributed control of a multirobot system. In *Proceedings of the 6th International Conference on Intelligent Autonomous Systems*, pages 115–122, 2000.

[5] J. Fredslund and M.J. Mataric. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, Oct. 2002.

[6] B.P. Gerkey and M.J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954, Sep. 2004.

[7] A. Howard, Parker, L.E., and G. Sukhatme. Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. *International Journal of Robotics Research*, 25:431–447, 2006.

[8] E.G. Jones, M.B. Dias, and A. Stentz. Time-extended multi-robot coordination for domains withintra-path constraints. *Autonomous Robots*, 30(1):41–56, 2011.

[9] N. Kalra, D. Ferguson, and A. Stentz. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[10] M. Lemay, F. Michaud, D. Letourneau, and J.M. Valin. Autonomous initialization of robot formations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 3018–3023, Apr. 2004.

[11] S. Liemhetcharat and M. Veloso. Weighted synergy graphs for role assignment in ad hoc heterogeneous robot teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5247–5254, 2012.

[12] P. Ogren and N.E. Leonard. Obstacle avoidance in formation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 2492–2497, Sep. 2003.

[13] Parker, L.E. and F. Tang. Building multirobot coalitions through automated task solution synthesis. *Proceedings of the IEEE*, 94(7):1289–1305, Jul. 2006.

[14] V. Sujan and S. Dubowsky. Visually guided cooperative robot actions based on information quality. *Autonomous Robots*, 19(1):89–110, Jul. 2005.

[15] L. Vig and J.A. Adams. Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22(4):637–649, 2006.

[16] Y. Zhang and Parker, L.E. IQ-ASyMTRe: Synthesizing coalition formation and execution for tightly-coupled multirobot tasks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[17] Y. Zhang and Parker, L.E. IQ-ASyMTRe: Forming executable coalitions for tightly coupled multirobot tasks. *IEEE Transactions on Robotics*, 29(2):400–416, 2013.

[18] Y. Zhang and Parker, L.E. Multi-robot task scheduling. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2992–2998, May 2013.