# IQ-ASyMTRe: Forming Executable Coalitions for Tightly Coupled Multirobot Tasks

Yu Zhang, *Member, IEEE*, and Lynne E. Parker, *Fellow, IEEE*

***Abstract*—While most previous research on forming coalitions mainly concentrates on loosely coupled multirobot tasks, a more challenging problem is to address tightly coupled multirobot tasks that involve close robot coordinations, which often require capability sharing. General methods for autonomous capability sharing have been shown to greatly improve the flexibility of distributed systems. However, in addition to the interaction constraints between the robots and the environment as required by the tasks, these methods may introduce additional interaction constraints between the robots based on how the capabilities are shared. The satisfiability of these constraints in the current situation determines the feasibility of the potential coalitions. To achieve system autonomy, the ability to identify the potential coalitions that are feasible for task execution is critical. In this paper, we demonstrate a general approach that incorporates this capability based on the ASyMTRe architecture. The extended architecture, which is called IQ-ASyMTRe, is able to find coalitions in which these required constraints are satisfied. When used to form coalitions, IQ-ASyMTRe sets up only feasible coalitions, thus enabling tasks to be executed autonomously. We formally present the new architecture and prove that it is sound and complete, given certain assumptions. Simulations and experimental results are provided for different applications in which the robots are able to flexibly form coalitions that are ready to execute.**

***Index Terms*—Coalition formation, distributed robot systems.**

## I. Introduction

MULTIROBOT tasks are those that require multiple robots to cooperate by forming subgroups (i.e., coalitions) to accomplish the given task. An intuitive approach for reasoning about forming coalitions to address a single multirobot task[1] is to divide the task into subtasks or roles that individual robots can perform. For example, in a robot insertion task [1], a supervisor robot would provide visual information to guide the implementor robot, which would execute the insertion. An issue with this approach, however, is that subtasks or roles have to be predefined by the human designer, which is not practical for arbitrary tasks. It is also desirable for the reasoning to be dynamically dependent on the capabilities (whether sensory, motor, or computational) of the available robots so that coalitions can be formed at a more fine-grained scale. Such an ability is especially required in tightly coupled multirobot tasks, which require robots to continually share capabilities during task execution. These situations commonly occur when it is impractical to install the required sensors on all (potentially heterogeneous) robots, or when using sensors positioned on different robots is more suitable in the current situation. Forming coalitions based on subtasks or roles is often not informative enough for autonomous capability sharing.

The representation of robot capabilities clearly has a critical influence on the design of such desired systems. First, the capabilities should be defined so that the systems can be implemented with different robots and applied to various tasks. Modularity is also important to make the systems easily extendable to different task domains, such as extending the robot insertion task to a robot box pushing task. Finally, it is necessary to define a uniform interface for the capabilities that models the interactions among the robots and the environment so that the robots can autonomously reason about how the capabilities must interact in the environment for a given task.

General methods that satisfy the independence and modularity requirements greatly increase system capability. Approaches that also enable autonomous reasoning with capability sharing [2], [3] have been shown to further improve the system flexibility in tightly coupled multirobot tasks. However, an important issue that remains unaddressed is that the constraints introduced by the required interactions are not considered; these constraints determine the feasibility of the potential coalitions. For example, a robot without a localization capability may need help from another robot in order to navigate. This, in turn, requires the robots to interact in close proximity since the relative position between the robots has to be retrieved using sensors of limited range (e.g., cameras). A coalition with robots that are not in each other's sensor field of view (FOV) is infeasible. It is clear that the feasibility of the coalitions is not only influenced by the capabilities of the robots, but also by the configuration of the robots (e.g., position) and the environment. Previous approaches to form coalitions do not consider this issue and, thus, cannot easily identify whether the formed coalitions are feasible for execution.

This paper presents the IQ-ASyMTRe architecture,[2] which, to the best of our knowledge, is the first attempt to address the

Manuscript received May 2, 2012; revised; accepted November 9, 2012. This paper was recommended for publication by Associate Editor J. A. Castellanos and Editor D. Fox upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation under Grant 0812117.

The authors are with the Department of Electrical Engineering and Computer Science, The University of Tennessee, Knoxville, TN 37996-3450 USA (e-mail: yzhang51@eecs.utk.edu; parker@eecs.utk.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2012.2228135

[1] In this paper, we consider addressing an individual multirobot task, which involves forming one or multiple coalitions.

[2] The architecture was initially presented in [4]. IQ is short for information quality. This paper extends the previous work to include formal analyses and more simulations and experiments.

issue of forming coalitions that are executable. Our approach can also be used in coalition execution to identify situations when the feasibility of the coalitions dynamically changes, requiring robots to adjust their behaviors. More discussions on dynamic execution can be found in [4]. After an overview of the related work (see Section II), we first provide some background knowledge (see Section III). The new IQ-ASyMTRe architecture is then explained (see Section IV). Afterward, we present simulations and experimental results (see Section V). Finally, we offer concluding remarks (see Section VI).

## II. Related Work

An extensive amount of work [5]–[13] has addressed multirobot cooperation with single-robot tasks, in which tasks are independent. Compared with this problem, addressing multirobot tasks is more complex. Approaches that divide multirobot tasks into subtasks or roles using domain knowledge [1], [14] effectively reduce them to single-robot tasks so that the previous work can easily apply. Variants of the contract net protocol [15] are often used to coordinate the robots [8], [13], [16]. However, these approaches can limit the capability of the systems in tightly coupled multirobot tasks, in which robots are required to share capabilities.

To reason about forming coalitions at a more fine-grained scale, researchers often adopt a numeric representation [17]–[21] for robot capabilities. The advantage is that this representation is more conveniently subject to theoretic analysis. The disadvantage, however, is that it does not facilitate autonomous capability sharing. Alternatively, a behavior-based representation, such as schema theory [22], [23], can be designed for modularity and provides an interface for the modules so that they can interact with each other, although the unstructured interface is too generic to enable autonomous reasoning. An advantage of this approach is that more complex behaviors can be achieved using the basic ones, enabling high-level decision-making architectures [12], [24]–[26] to be applied.

Building upon schema theory, the ASyMTRe [2] architecture was introduced to address tightly coupled multirobot tasks; it is also the first architecture to enable autonomous capability sharing at the sensory and computational levels. For a more structured interface, ASyMTRe uses *information type*[3] as input and output labels of schemas (i.e., modules) to attach information with semantic meanings. Schemas can only be activated when their input information types are satisfied; they also produce specified output information types. This way, ASyMTRe is able to autonomously reason about the interactions between the schemas based on how information should flow within the distributed system to where it is required. Capability sharing is implicitly achieved through the communication of information. Unlike sensor fusion architectures [27], [28] that address the problem of combining sensory information from different robots, ASyMTRe is designed to model the information flow among very different modules (e.g., motors and sensors).

[3]While *data type* refers to the format of the data (i.e., integer), *information type* defines the semantic meaning of the data (e.g., global or relative position).

While other approaches [17], [29], [30] to address tightly coupled multirobot tasks exist, they do not enable autonomous capability sharing. Some of them are designed to work at the robot level. As a result, coordination between the robots is implemented in a task-dependent manner; these solutions cannot be easily extended to different task domains. For example, the work in [29] introduces a market-based framework for tight coordination in a security sweep task domain. Passive coordination is used to quickly produce solutions for local robots, while active coordination is used to produce complex solutions via coordination between teammates. Other approaches [17], [30] that model robot capabilities, however, do not facilitate autonomous capability sharing, due to the lack of a structured interface. Unlike in the multiagent domain, sensory capabilities are located on different robots and cannot be easily transferred. Vig and Adams [30] first noted this, and addressed it by restricting coalitions to those that satisfy the location constraints of the sensors. A better solution is for the robots to autonomously share capabilities as enabled in [2] and [3], such that these constraints are relaxed.

Meanwhile, although application-specific methods for capability sharing [1], [8], [14], [31] can often be used (by specifying how each agent or robot interacts with others to share information), such an approach is often restricted to work with specific systems, in which the solutions for the interactions are easy to characterize. As a result, these methods cannot generalize to scenarios that require complex interactions with dynamic systems. Hence, we believe that a general architecture is required and that a schema-based design, which is coupled with a structured interface based on information types, satisfies the requirements to address tightly coupled multirobot tasks. However, while this approach enables us to reason about the required interactions for creating potential coalitions, it cannot determine which coalitions are feasible for execution in the current situation, since the constraints that are introduced by these interactions are not considered. The information type (to specify the interactions) is statically defined with respect to the capabilities of schemas and does not include information on how the coalitions are being instantiated in the environment. This paper addresses these issues, enabling the formation of executable coalitions in the current situation.

## III. ASyMTRe

Based on schema theory [22], [23], the ASyMTRe architecture [2] defines basic building blocks of robot capabilities to be collections of environmental sensors (ESs), perceptual schemas (PSs), motor schemas (MSs), and communication schemas (CSs). A set of information types $F$s is introduced to label the inputs and outputs of schemas. Connections between schemas can be made when the output label of one matches the input label of another. To reason about coalition solutions, ASyMTRe searches through all possible ways to connect different schemas in order to activate the required MSs on the robots to achieve a task. The activation of each schema is associated with a cost, which is used in ASyMTRe to compare alternative solutions. ASyMTRe uses an anytime algorithm with heuristics

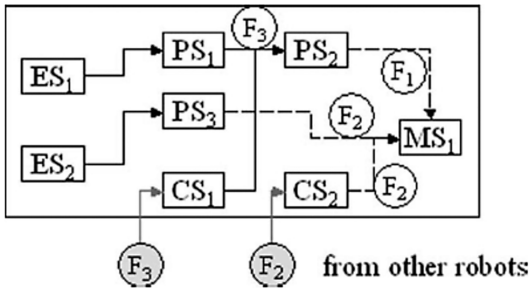



Fig. 1. Example of how schemas are connected in ASyMTRe [2].

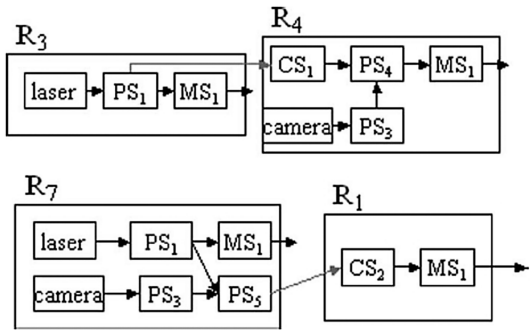



Fig. 2. Retrieving the same information in different ways in ASyMTRe [2].

to return good solutions earlier. Fig. 1 illustrates an example of schema connections found by ASyMTRe. While a solid line represents an *AND* condition, dashed lines represent *OR*. For example, the figure shows that the requirement of information type $F_2$ can be provided by either $\mathrm{PS}_3$ or $\mathrm{CS}_2$ (see [2] for more details).

While ASyMTRe and other architectures that are based on a similar idea (see, e.g., [3]) have been shown to improve the flexibility of multirobot systems, they all suffer from several issues. First, the definition of information type limits the referenced information to only be statically dependent on the capabilities of the schemas, instead of dynamically dependent on the actual information retrieved. This can cause problems when sensors that are capable of producing an information type cannot retrieve the required information in the current situation. For example, in a navigation task, a robot without a localization capability, and a robot with this capability, may not always be within each other's FOV. In such cases, these architectures may choose to form infeasible coalitions. To address this issue, schemas must be activated in the planning phase when necessary. The incorporation of such a process is missing in ASyMTRe and these previous architectures for forming coalitions. Furthermore, PSs (or a specific type of *action* in [3]) can be dependent on how the input information is retrieved, which requires application-specific code to be designed. Fig. 2 shows two scenarios for how robots without a localization capability ($R_4$ and $R_1$ in the figure) retrieve their global position information with help from another robot. In the top scenario, position information relative to $R_3$ (with a laser-based localization capability) is retrieved by $R_4$ using a camera, whereas in the bottom scenario, the relative information is retrieved by $R_7$. These previous shortcomings are addressed by IQ-ASyMTRe, which is presented next.

## IV. IQ-ASyMTRe Architecture

We begin by first extending the representation of information to introduce a complete definition of information type that includes referent information. This extended representation is needed to dynamically reason about the feasibility of coalitions. A new type of PS, which is called *reduction PS*, is also introduced to model the conversions between information types. We then present the new solution space and potential solutions which result from these extensions and discuss how they relate to the coalition solutions (to specify the coalitions and the required interactions). Finally, the IQ-ASyMTRe algorithm is presented; its properties are discussed and proven afterward.

### A. Information Type and Information Instance

The incompleteness of *information type*, as originally defined in [2], is due to the fact that the relationships between entities[4] and information are not explicitly captured. Intuitively, information must be specified with a set of referents. For example, the information of $r_A$*'s global position* is not useful without specifying $r_A$. IQ-ASyMTRe uses both *information type* and *information instance* for a complete definition.

*Definition 4.1 (Information Type):* An information type in IQ-ASyMTRe is specified by a pair ($\mathcal{F}$, $N$), where $\mathcal{F}$ is a label for the semantic meaning of the information that defines the specific sensing or computational data of a schema or a sensor. $\mathcal{F}$ is consistent with the definition of information type in ASyMTRe, while $N$ is the number of referents that should be associated with $\mathcal{F}$. □

For example, the information type of global position can be specified by ($\mathcal{F}_G$, 1). This information type has only one referent, which refers to one entity's global position. In order to specify the complete semantic meaning, however, information type alone is not sufficient. As an example, ($\mathcal{F}_G$, 1) does not inform us about whose global position it is. To address this issue, we introduce the definition of *information instance*.

*Definition 4.2 (Information Instance):* An information instance in IQ-ASyMTRe not only contains the semantic meaning expressed in its information type, but it also captures information about the referents. An information instance of a particular information type ($\mathcal{F}$, $N$) can be represented as $F(\mathit{Ref}_{1:N})$, where $\mathit{Ref}_j$ is the $j$th referent for $F(\mathit{Ref}_{1:N})$. □

Each referent $\mathit{Ref}_j$ can be instantiated to a particular entity or remain uninstantiated for future instantiation. Fully instantiated information instances represent actual information that can be used. Partially instantiated information instances represent a class of information. For example, $F_G(X)$ can be the global position information of any entity to which $X$ is instantiated. The use of information instance creates a complete reference of information. For example, after instantiating the two referents of an information instance of the relative position information type ($\mathcal{F}_R$, 2) to robots $r_A$ and $r_B$, respectively, the reference of information $F_R(r_A, r_B)$ has a unique meaning (i.e., $r_A$'s position

[4]Entities can be locations, robots, agents, or objects that can be identified in the environment.

relative to $r_B$), no matter how the information is retrieved or used.

The definition of a complete reference of information enables dynamically reason about coalition solutions, based on how capabilities or information is shared, such that infeasible coalitions for execution can be identified. To achieve this dynamism, statically labeling the schemas with information types as in ASyMTRe is insufficient; instead, in IQ-ASyMTRe, the semantic labels (information instances) are also encoded in the information flowing through the schemas. For example, for sensory information retrieved by an ES, an associated PS extracts the semantic information in order to dynamically instantiate the referents of its static label; it then outputs the new label along with the actual data. In such a way, IQ-ASyMTRe can use actual sensory or communicated information to form coalitions, making it more powerful compared with other approaches.

We further introduce the concept of *generality* for information instances for later use. In the rest of this paper, referents for information instances are not shown unless necessary.

*Definition 4.3 (Generality of Information Instance):* For two information instances ($F^x$ and $F^y$) of the same type ($\mathcal{F}$, $N$), $F^x$ is more general than $F^y$ (denoted by $F^x \succ F^y$), if and only if the following two statements are true.

1) $\exists$ $Ref_k \in Ref_{1:N}$ in $F^y$ that is instantiated while the corresponding referent ($Ref_k$) in $F^x$ is not.
2) $\forall$ $Ref_k \in Ref_{1:N}$ in $F^x$ that is instantiated, the corresponding referent ($Ref_k$) in $F^y$ is also instantiated.

In addition, $F^x$ is as general, or has the same generality, as $F^y$ (denoted by $F^x = F^y$), if and only if the following two statements are true.

1) $\forall$ $Ref_k \in Ref_{1:N}$ in $F^y$ that is instantiated, the corresponding referent ($Ref_k$) in $F^x$ is also instantiated.
2) $\forall$ $Ref_k \in Ref_{1:N}$ in $F^x$ that is not instantiated, the corresponding referent ($Ref_k$) in $F^y$ is also not instantiated.

□

Note that Definition 4.3 only specifies partial orders for information instances of the same type in IQ-ASyMTRe, since two information instances of the same type may or may not be comparable. For example, given that $F^x \not\succ F^y$ and $F^x \neq F^y$, it is not necessarily true that $F^y \succ F^x$.

### B. Information Conversion (Reduction PS)

The Reduction PS (denoted by *RPS* henceforth) is introduced in IQ-ASyMTRe to express information conversions. The idea is to provide a constructive way to reason about the relationships between different information systems, as first introduced in information invariants theory [32]. This capability is desirable to reason about forming coalitions, since potential coalition solutions represent ways to connect different components (i.e., schemas) to form equivalent information systems (i.e., to retrieve the required information). Since information conversions are general, no application-specific code needs to be designed for the reasoning process.

Furthermore, combined with a complete reference of information, IQ-ASyMTRe can express information conversions in which multiple information instances of the same type are used; such flexibility is not accessible to architectures that are based solely on information types. For example, from the relative position of robot $r_B$ to robot $r_A$ and robot $r_C$ to robot $r_A$, one can compute the relative position of $r_B$ to $r_C$.

Information conversions that are expressed in IQ-ASyMTRe can be specified in the Backus–Naur form (BNF) as follows.

*Definition 4.4 (Information Conversion):* Information conversions in IQ-ASyMTRe express relationships between composite information instances (abbreviated as comp inst in the BNF specifications). □

A composite information instance is constructed from connected information instances using logic operators {*iAND*, *iOR*}, which are similar to {*AND*, *OR*} in propositional logic, such that the distributive property also holds. Information conversions convert the composite information instance on the left-hand side to the one on the right:

$$< \textit{info conversion} > ::= < \textit{l-comp inst} > \Rightarrow < \textit{r-comp inst} > .$$

The BNF of a composite information instance is given as follows, in which *information instance* is abbreviated as *info inst*. Since *iOR* operators on the right-hand side are not well defined[5], the definitions for the two sides are different:

$$\begin{aligned} < \textit{l-comp inst} > ::= & (< \textit{l-comp inst} > \textit{iAND} < \textit{l-comp inst} >) \\ & \mid (< \textit{l-comp inst} > \textit{iOR} < \textit{l-comp inst} >) \\ & \mid < \textit{info inst} > \\ < \textit{r-comp inst} > ::= & (< \textit{r-comp inst} > \textit{iAND} < \textit{r-comp inst} >) \\ & \mid < \textit{info inst} > . \end{aligned}$$

*Lemma 4.1:* Information conversions in IQ-ASyMTRe can always be defined with one or multiple information instances connected with only *iAND* operators on the left-hand side and a single converted information instance on the right.

*Proof:* First, since information instances on the right are connected with only *iAND* operators, we can easily divide any information conversions in Definition 4.4 into multiple conversions with a single information instance on the right. This is done by creating a separate conversion for each information instance on the right with the left-hand side unchanged. Afterward, by using the distributive property, we can transform the composite information instances on the left into their respective disjunctive normal forms (DNFs). Finally, we simply need to divide each of the transformed information conversions into multiple conversions by introducing a new conversion for each conjunctive clause on the left. ■

As an example, for an information conversion having the form ($A$ *iOR* $B$) *iAND* $C \Rightarrow D$ *iAND* $E$, we can first divide it into two information conversions for each information instance on the right-hand side. The two conversions are ($A$ *iOR* $B$) *iAND* $C \Rightarrow D$, and ($A$ *iOR* $B$) *iAND* $C \Rightarrow E$. Next, we apply the distributive rule to the first conversion to get ($A$ *iAND* $C$) *iOR*

[5]For example, saying that the information instance of $A$ can be converted to $B$ or $C$ implies that both $B$ and $C$ can be converted; therefore, *iAND* should be used instead of *iOR*.

TABLE I
EXAMPLES OF RPS'S USED IN IQ-ASYMTRE

| 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 |

$(B$ *iAND* $C) \Rightarrow D$. The second one can be transformed similarly. Finally, by introducing a conversion for each conjunctive clause, we have $A$ *iAND* $C \Rightarrow D$, and $B$ *iAND* $C \Rightarrow D$.

*Definition 4.5 (IQ Information Conversion):* Lemma 4.1 allows us to express any valid information conversion in the following form (called the "IQ form" henceforth):

$$< \textit{info conversion} > ::= < \textit{l-comp inst} > \Rightarrow < \textit{r-comp inst} >$$
$$< \textit{l-comp inst} > ::= (< \textit{l-comp inst} > \textit{iAND} < \textit{l-comp inst} >) \mid < \textit{info inst} >$$
$$< \textit{r-comp inst} > ::= < \textit{info inst} > . \qquad \square$$

The advantage of defining information conversions in the IQ form is that the reasoning process to create the solution spaces is significantly simplified. In the following discussions, we assume that all information conversions are defined in the IQ form; we also denote the *iAND* operator by "+," for conciseness. Table I shows several RPSs that can be used in IQ-ASyMTRe. They are general since the referents can be instantiated to different entities, as long as the same referent labels are instantiated to the same entities.

### C. Solution Space and Potential Solution

*Potential solutions* represent the possible alternative ways that schemas can be connected on a robot to retrieve the required information for the task; a *solution space* is created on this robot to encode all such potential solutions. Although the introduction of information instance and RPS significantly changes the solution space and potential solutions in IQ-ASyMTRe, the reasoning process to create them remains similar to [2]. To create the solution space, the reasoning algorithm checks all schemas that can output the required information instances and then checks recursively for the inputs of those schemas until the path either ends in a conflict with the *referent instantiation constraint*[6] or in a terminal state (i.e., CS or ES–EPS pair[7] that can be the source of the required information instance). Note that while the reasoning to create the solution space is similar to STRIPS [24] planning (considering sets of information instances as states and RPSs as reduction rules), IQ-ASyMTRe provides a more manageable reasoning system for the problems that we address by restricting the spaces of states and rules.

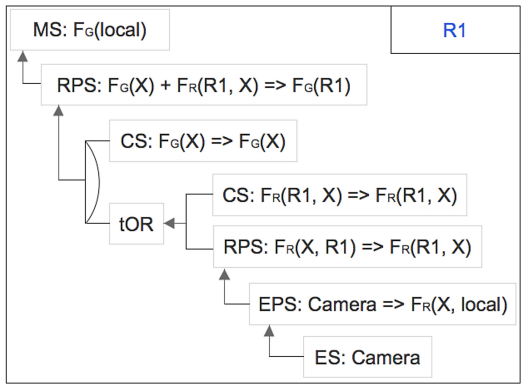



Fig. 3. Solution space in the cooperative robot navigation task [2] for a robot to obtain its global position with only a camera sensor. The referent *local* refers to the robot itself. This solution space encodes two potential solutions. One solution is to have another robot $X$ send over its global position (CS: $F_G(X) \Rightarrow F_G(X)$) and use the camera sensor to sense the relative position (EPS: Camera $\Rightarrow F_R(X, \text{local})$). An RPS (PS: $F_R(Y,X) \Rightarrow F_R(X,Y)$) 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 $X$.

A solution space can be represented in a directed graph representation as an *and–or* tree (e.g., see Fig. 3). Each node in the solution space represents a schema or an ES–EPS pair. Each edge represents an information instance that constantly flows from the output of one node into the input of another (except for the edges connecting ESs and EPSs). The root of the *and–or* tree, which specifies the required information instances, can either be an MS or a CS; it acts as a sink node that information flows into. Since we assume that RPSs are defined in the IQ form, every node has one or more incoming edges and a single outgoing edge. Every leaf node (i.e., the information source) is either a CS, representing information communicated from others, or an ES–EPS pair, representing information retrieved using sensors. Nodes that are closer to the sink node (i.e., the root) are said to be downstream of the nodes further away on the same branching path,[8] as information flows from the leaves to the root. Fig. 3 shows a solution space for the cooperative robot navigation task [2]. The *tOR* node is introduced to manage multiple options of connection. Another example that involves different information types is provided in Fig. 4; it shows the box pushing task presented in [32], which uses force feedback, infrared, bumper, and position sensors. A cooperative robot transportation task that requires robots to be physically connected can be similarly represented. These examples show the applicability of IQ-ASyMTRe to varying task domains.

After the solution space is created, potential solutions can be extracted from the root to the leaves by making decisions on which schema node to use at each *tOR* node; the rest of the nodes are trimmed. Note that after the selections for the nodes are made, the *tOR* nodes are no longer necessary and can be removed for a clearer representation. The cost to use a potential

[6]The referent instantiation constraint requires the same labels to be instantiated to the same entities in the inputs and outputs of the same schemas. Validation of this constraint occurs in both of the algorithms presented in Section IV-G. In the algorithm to create the solution space, it is used to remove invalid potential solutions, while it is used in the second algorithm to determine the feasibility of the coalitions.

[7]Sensory information instances are extracted from raw sensor data using PSs that are designed for the specific ESs. PSs of this kind are denoted by EPSs in the following discussions.

[8]A branching path is the path that starts from any leaf node and follows the information flow until the root node (i.e., the sink node) is reached.

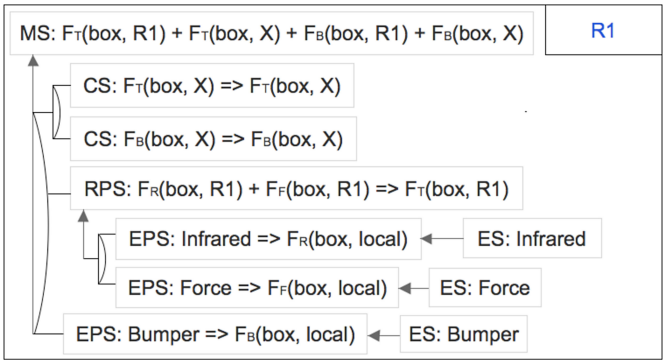



Fig. 4. Solution space that represents one of the protocols presented in [32] for two robots to cooperatively push a box in a given direction. For the robots to coordinate actions, they must estimate the net torque on the box based on the torque information $F_T$ and use the bumper information $F_B$ to determine whether the robots are in contact with the box. The torque for each robot is computed from its exerted force on the box $F_F$ and its relative position to the box $F_R$. Other protocols that were presented in [32] (e.g., using the offsets from the robots' starting positions) can be similarly represented.

solution can be computed as the sum of the costs of all schemas in the solution. Clearly, potential solutions also have *and–or* tree representations.

However, one issue with IQ-ASyMTRe is that the solution space can be of infinite size without restricting the use of RPSs. This is due to the fact that the same RPSs can potentially be used an unlimited number of times. To address this issue, we introduce the *Generality Imposition* constraint as follows.

1) *Generality Imposition* constraint—For each RPS, prohibit the use of the same RPS to convert information instances of the same type that have less or equal generality upstream on the same branching path.

*Lemma 4.2:* The *Generality Imposition* constraint ensures that solution spaces and potential solutions have finite graphical representations.

*Proof:* The representation of a solution space can become infinite when loops occur. This occurs when information instances of the same type that have the same generality with the same entity instantiations appear more than once on the same branching path of the solution space as outputs of the same RPS. The *Generality Imposition* constraint directly prohibits these occurrences of loops. Otherwise, we note that the numbers of RPSs, information types, and referents which are associated with information types are finite given a task domain. Hence, in the worst case, every branching path would terminate after information instances of all types reach the most general forms (i.e., have no instantiated referents). Hence, solution spaces have finite representations. As potential solutions are trimmed solution spaces, the conclusion is straightforward. ■

An additional conclusion is that the sizes of solution spaces (i.e., number of potential solutions) are also finite. To further reduce their sizes, we introduce two more constraints.

1) *Localness in Reasoning* constraint—No schema connection except for CS should be created if none of the referents for the information instance to be provided is instantiated to the local entity (except for those with all referents instantiated to nonrobot entities).[9]

Given this constraint, for example, $r_A$ would not directly provide $F_R(r_B, r_C)$, even though $r_A$ can compute it from $F_R(r_B, r_A)$ and $F_R(r_C, r_A)$. However, the information required for the computation is available upon request.

1) *External Communication* constraint—CSs are used only when some referents are not instantiated to the local entity.

For example, if $r_A$ needs $F_G(r_A)$, it cannot request it directly. Instead, it must first seek other ways to obtain the information (e.g., computing it from $F_G(X)$ and $F_R(X, r_A)$). In this manner, computation load is also distributed. Note that $r_A$ may request information from itself.[10]

### D. Coalition and Coalition Solution

In IQ-ASyMTRe, potential solutions only specify how the *local robot*[11] $r_L$ directly interacts with others. To make this difference for later discussions, we define *local coalitions* differently from traditional *coalitions* [33].

*Definition 4.6 (Local Coalition):* A local coalition defined in IQ-ASyMTRe consists of $r_L$ and the robot teammates that directly feed information to $r_L$ to activate a required MS for the task or a CS for helping other robots. □

To search for local coalitions, IQ-ASyMTRe checks potential solutions in the solution space in a nondescending order based on the costs. For each potential solution, IQ-ASyMTRe activates the ES–EPS pairs to retrieve sensory information, and/or the CSs to send out information requests, as specified in the potential solution. ES–EPS pairs are only activated temporarily, and the information requests are sent only once (or a few times when communication links are unreliable). Robots that receive an information request create a solution space for the required information and follow the same search process for this space. If the information is retrievable, these robots keep sending the information until the temporary activations of the schemas expire. IQ-ASyMTRe uses the collected information to instantiate the required information instances in the potential solution and perform validation of the referent instantiation constraints. The resulting local coalition is feasible only if all such constraints are satisfied. A chosen feasible local coalition can then be set up by sending coalition requests to the relevant robot members. The instantiated potential solution is referred to as a *local coalition solution*, which is defined as follows.

*Definition 4.7 (Local Coalition Solution):* A local coalition solution (LCS) for a local coalition is the potential solution (selected by $r_L$) after fully instantiating all information instances. □

[9]For a robot to reason about information for helping others upon request, when there is no referent instantiated to the local entity but there are uninstantiated referents, the robot checks all possible ways to instantiate one of them to the local entity.

[10]This is a technical detail needed to prove completeness. However, in practice, less-costly alternative solutions often exist that do not involve self-communication. Note that self-communication can always be avoided by optimizing the implementation.

[11]The *local robot* refers to the specific robot that initiates the coalition.

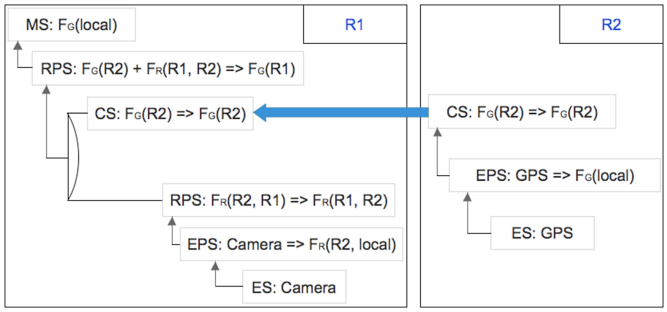



Fig. 5. Possible coalition solution for a potential solution in Fig. 3, in which the left part corresponds to the LCS.

Note that a potential solution can be instantiated to different LCSs. The left part of Fig. 5 shows the LCS for a valid instantiation of a potential solution in Fig. 3. To define *coalitions*, note that robots in the local coalition may often be interacting with others in order to retrieve the requested information for $r_L$. For example, in Fig. 3, $F_G(X)$ may be retrieved by $X$ with help from another robot.

*Definition 4.8 (Coalition):* A coalition includes all robots in the local coalition and robots that indirectly support $r_L$. □

A robot that directly feeds information to $r_L$ (i.e., in the local coalition for $r_L$) might also be using a local coalition to retrieve the necessary information, which introduces another coalition (which could be of size 1) for this robot. Such a process can continue recursively. That is, a coalition may include (or require) other coalitions, which may recursively include yet others. Hence, recursively, the feasibility of a coalition can only be determined when the feasibility of these required coalitions is determined; a coalition can only be set up after these required coalitions are set up. While the cost of a local coalition is that of the corresponding potential solution, the coalition may incur additional cost due to these required coalitions. The *coalition cost* is the sum of all these costs.

*Definition 4.9 (Coalition Solution):* The coalition solution for a coalition includes the LCS, as well as solutions for fulfilling the required coalitions recursively. □

Coalition solutions also have *and–or* tree representations since they are created by connecting LCSs, which are *and–or* trees. One characteristic of the coalition solutions, which differs from LCSs, is that the leaves always represent the ultimate information sources, i.e., ES–EPS nodes. Note that dummy sensors can be created to provide any prior information when necessary. Fig. 5 shows a coalition solution that instantiates a potential solution in Fig. 3.

Although the *Generality Imposition* constraint that was introduced in the previous section prevents loops in the potential solutions (and hence in the LCSs), it does not prevent loops in the coalition solutions. For example, while $r_A$ is requesting some information from $r_B$, $r_B$ may in turn request necessary information from $r_A$ to retrieve the requested information, which can lead to $r_A$ requesting the same information from $r_B$ again. To address this issue, we introduce the *Distinct Requests* constraint as follows.

1) *Distinct Requests* constraint—For any robot, prohibit the use of CS if the robot has already used CSs to request the same information instance.

*Lemma 4.3:* Given the *Generality Imposition* and the *Distinct Requests* constraints, LCSs and coalition solutions have finite graphical representations.

*Proof:* As LCSs are instantiations of potential solutions, given *Lemma 4.2*, the conclusion for LCSs is straightforward. The representation of coalition solutions can become infinite when loops occur; this occurs only when one robot is requesting an information instance, which ultimately leads to itself requesting the same information again. The *Distinct Requests* constraint directly prevents loops from occurring. Hence, coalition solutions also have finite representations. ■

Lemma 4.3 implies that any information request is populated only a finite number of times in the distributed system, although the information requested during the population may not necessarily be the same as in the initial request.

### E. Completeness of Solution Space

An important question is whether the solution spaces are influenced by the introduced constraints. Interestingly, the following lemma shows that the solution spaces are still complete in IQ-ASyMTRe.

*Lemma 4.4:* The combination of the following four constraints does not influence the completeness of the solution space for the distributed system.

1) the *Generality Imposition (GI)* constraint;
2) the *Localness in Reasoning (LR)* constraint;
3) the *External Communication (EC)* constraint;
4) the *Distinct Requests (DR)* constraint.

*Proof:* By definition, a coalition solution has an *and–or* tree representation. Furthermore, the leaf nodes, which are located on robots within the coalition, are ES–EPS nodes to retrieve the required sensory information instances, which flow within and across robots through schemas to the root (i.e., the sink node) of the coalition solution.

With no constraint, the simplest solution is to have sensory information instances from other robots sent directly to $r_L$ (i.e., where the root is located) and move RPSs to process them to $r_L$. Since no constraints are imposed here, self-communicating CSs on $r_L$ are unnecessary and can be easily removed via concatenation. All information instances are fully instantiated in this solution. The goal is to show how this simple solution can be reconstructed equivalently (via pruning and grafting) on the robots so that the potential solutions before the instantiations satisfy all constraints.

We start thinking in a reverse order; that is, given the created simple coalition solution, what could the potential solutions have looked like? First, note that potential solutions are LCSs before the instantiations and that the LCSs are part of the coalition solutions. Hence, any referent that is not instantiated statically (i.e., referents with instantiations specified *a priori* by the task or the EPSs) would not be instantiated in the potential solutions. After removing all nonstatic instantiations, we next show how

to reconstruct the uninstantiated coalition solution to resolve all possible violations.

First, for the ES–EPS nodes that remain on other robots, observe that for most sensors, the retrievable information instances are always related to the robots on which the sensors are located. Based on this observation, these information instances must have a referent instantiated to the respective local robot entities. Hence, the schema connections do not violate the *LR* constraint. It is easy to verify that they also do not violate the other constraints, since the node (schema) types do not apply.

Now, we can concentrate on the LCS for $r_L$. We start the process in a reverse breadth search fashion (i.e., from deeper to shallower nodes), until we encounter a violation at a node $v$. In the following, we use *Down*($x$) to denote downstream nodes of $x$ on the same branching path and $F_x$ to denote the information instance produced by node $x$. We also use *Subtree*($x$) to denote the upstream subtree rooted at $x$ and CS($F$) to denote a CS for requesting $F$. There are four possible situations.

a) In the case of a violation of the *DR* constraint, without changing anything, we can simply implement the communication module in such a way that for all CSs on a robot that request the same information instance, only the first CS created may send the request. The intuition is simple: There is no need to make duplicate requests.

b) We can argue that the case of a violation of the *EC* constraint would not occur. If it does occur, we know that $v$ is a CS and the only referent in $F_v$ is statically instantiated to $r_L$, since instantiating two or more referents of an information instance to the same entity would not be informative by definition. Hence, $F_v$ must not be communicated from other robots based on our previous observations (or there should be instantiations with other robot entities). Consequently, $v$ should be an ES–EPS or an RPS node, instead of a CS node, according to the construction of the simple coalition solution. Note also that the resolution process for the violations of other constraints does not introduce this type of violation.

c) In the case of a violation of the *GI* constraint, we know that $\exists u \in$ *Down*($v$), for which $F_u \succ F_v$ or $F_u = F_v$. Furthermore, given the search order, we also know that *Subtree*($v$) contains no violations on $r_L$.

c.1) If there exists a referent in $F_v$ that is instantiated to a robot entity in the coalition, which is denoted by $r_E$ ($r_E \neq r_L$), we can trim off *Subtree*($v$) and replace it with CS($F_v$). The creation of the CS does not violate any constraints; furthermore, *Subtree*($v$) can be moved to $r_E$ to reason about $F_v$ incurring no violations by using the following process.

For leaf nodes that are ES–EPS nodes in *Subtree(v)*, we need to replace them with CSs. This occurs after moving *Subtree(v)* to $r_E$ to request the sensory information instances from $r_L$. For the rest of the leaf nodes (CSs), if the transferred information is not from $r_E$, we need not do anything. Otherwise, the process is more complicated. In the following, we denote the information that is transferred from $r_E$ by $F_E$, and the node that is immediately upstream of CS($F_E$) on $r_E$ by $E$. If $F_E$ has more than one referent and one of them is instantiated to $r_E$, we need not do anything. Otherwise, we have two cases.

c.1.1) $F_E$ has only one referent: If the only referent is not instantiated to $r_E$, the situation falls into the second case (i.e., c.1.2); else, if the only referent is not statically instantiated to $r_E$, nothing needs to be done; else, the only referent is statically instantiated to $r_E$. Since instantiations are only inherited upstream until reaching the ES–EPS leaf nodes, $F_v$ must have a referent statically instantiated to $r_E$ as well.

If $F_v$ has only one referent, then following the same referent inheritance property $F_u$ too has a referent statically instantiated to $r_E$. According to the definition of the *GI* constraint, we must have that $F_v$ and $F_u$ represent the same information. As a result, we can move *Subtree(v)* back to $r_L$ to replace *Subtree(u)*. Otherwise, $F_v$ has more than one referent. In that case, if $E$ is an ES–EPS node, we can remove the leaf node CS($F_E$) from *Subtree(v)* and concatenate it with *Subtree(E)* on $r_E$; else, we know that *Subtree(E)* must have only violated the *LR* constraint before being moved from $r_L$ to $r_E$, since it clearly did not violate the *EC* constraint. Furthermore, it must not have violated the *GI* constraint, since otherwise, *Subtree(E)* should have been used to replace the node in violation on $r_L$ (assuming that the resolution of the *GI* constraint takes precedence over that of the *LR* constraint). In such a case, we can simply remove the leaf node CS($F_E$) from *Subtree(v)* and concatenate it with *Subtree(E)*.

c.1.2) No referent of $F_E$ is instantiated to $r_E$: We can argue that this case would not occur. First, $E$ cannot be an ES–EPS node; otherwise, $r_E$ should be present. Meanwhile, $F_E$ must not have been reasoned out using an RPS either, since otherwise, *Subtree(E)* should not have been moved to $r_E$ due to a violation with the *LR* constraint.

c.2) Following c.1, if such a referent does not exist, there are three cases: 1) $F_v$ has more than one referent, and one of them is instantiated to $r_L$: In such a case, *Subtree(v)* can be replaced by a self-communicating CS for $F_v$; 2) $F_v$ has only one referent: If the only referent is not instantiated to $r_L$, the situation falls into the third case; else, if the only referent is not statically instantiated to $r_L$, *Subtree(v)* can be replaced by a self-communicating CS for $F_v$; else, the only referent is statically instantiated to $r_L$. It follows that $F_v$ and $F_u$ are the same; therefore, we can replace *Subtree(u)* with *Subtree(v)*. 3) No referent of $F_v$ is instantiated to $r_L$: In such a case, all referents in $F_v$ are statically instantiated to nonrobot entities, since these entities must be specified *a priori* by the task. Given that fully instantiated information instances of the same type with the same set of entities are almost always equivalent (i.e., there exists an RPS that can convert one to the other), we can use such an RPS to convert $F_v$ to $F_u$ and then replace *Subtree(u)* with the modified *Subtree(v)*. New *GI* violations that are introduced on the modified subtree can be resolved in a similar manner.

d) In the case of a violation of the *LR* constraint, we know that none of the referents of $F_v$ is statically instantiated to $r_L$. We can apply a similar process as in (c).

We can iterate the aforementioned process until all violations are resolved. On termination, we have created an equivalent coalition solution in which the potential solutions satisfy all of our constraints. Hence, the conclusion holds. ■

An example scenario for the navigation task is given in Fig. 6(a) in which four robots must activate the same MS to go to the same global position. The robots are positioned in a column formation, as shown in the figure. The simple coalition solution for the example scenario is shown in Fig. 6(b) for the

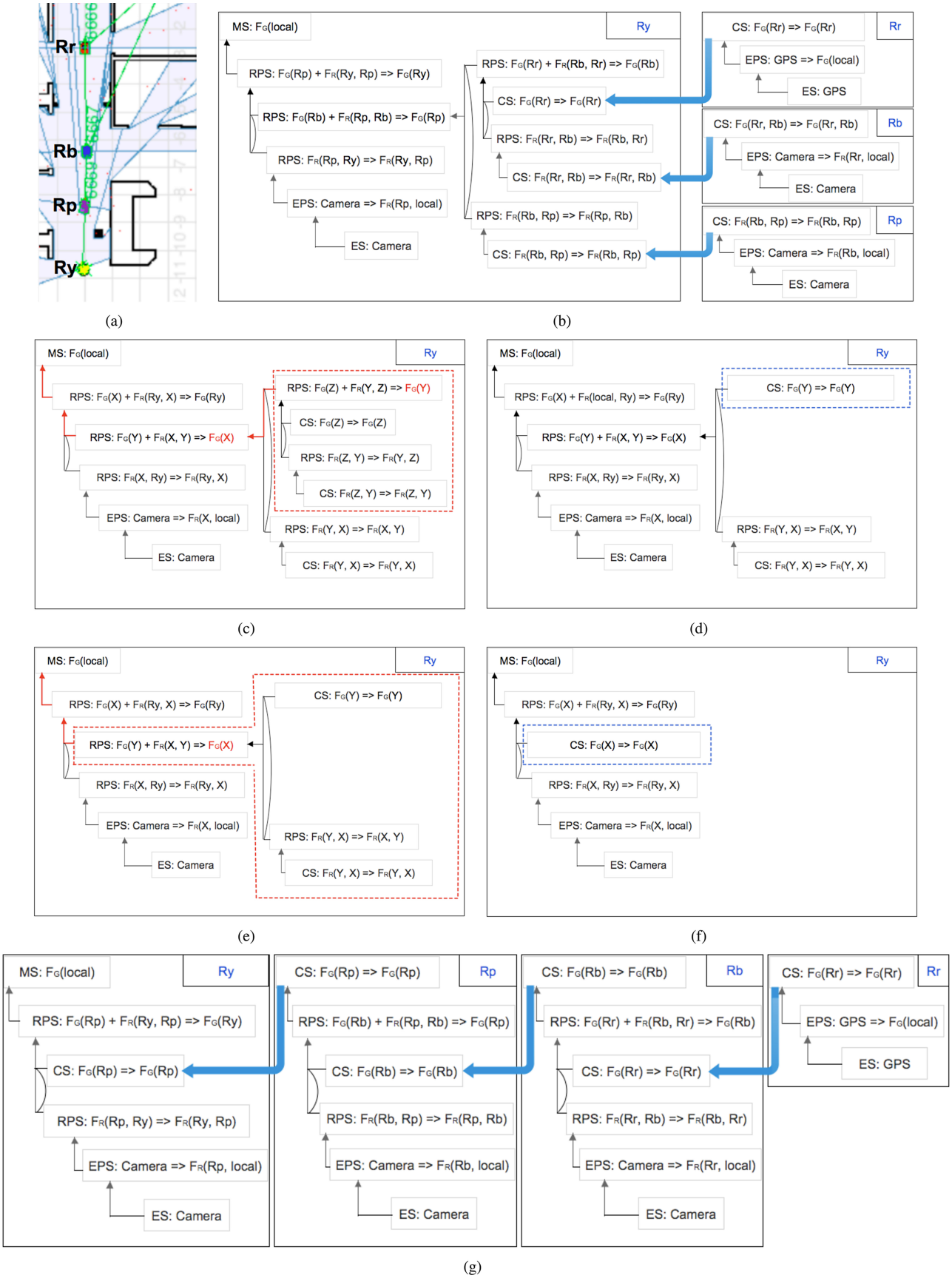



Fig. 6. (a) Scenario for the navigation task in which only the red robot (at the top) has a GPS for global positioning. The other robots only have a camera for relative positioning, and each can only see the robot immediately in front of it, since they are in a column formation. (b) Simple coalition solution for the scenario in (a). (c) LCS before instantiations. (d) After resolving the violation of the *GI* constraint. (e) Violation of the *LR* constraint. (f) After resolving the violation of the *LR* constraint. (g) Possible coalition solution after applying the potential solution in (f).

**Algorithm 1** A Recursive Algorithm for the Solution Space

```
for all input information instances of the current node do
  if LR is satisfied then
    for all EPSs that can retrieve the instance do
      if no conflict occurs between the information label
      and the EPS label then
        Create an ES−EPS node to expand the tree.
      end if
    end for
    for all RPSs that can produce the instance and satisfy
    the GI constraint do
      Populate the instantiations in the RPS based on the
      information label.
      Create a RPS node to expand the tree and pass on
      the new labels to recursively invoke the algorithm
      on the new node.
    end for
  end if
  if EC is satisfied then
    Create a CS node with the information label to expand
    the tree.
  end if
end for
return The current node after the expansions.
```

**Algorithm 2** The Algorithm for IQ-ASyMTRe

```
Invoke Algorithm 1 on a MS or CS with the required inputs.
Extract and order potential solutions (PoSs) in a list based
on predefined costs for schemas.
while true do
  if not all PoSs are checked then
    Retrieve the next PoS on the ordered list.
    Activate the required ES−EPS nodes temporarily in the
    PoS to collect sensory information.
    Activate CS nodes in the PoS to send information
    requests for the attached information labels.
  end if
  Collect information communicated by other robots.
  Process requests and share information.
  for all checked PoSs do
    if the required information is available and no conflict
    with the referent instantiation constraints occurs then
      Compute the coalition cost and record the coalition.
    end if
  end for
  if the given search time has elapsed then
    Negotiate to set up the coalition (with other robots)
    with the least coalition cost.
    return The generated coalition.
  end if
end while
```

bottom robot. Fig. 6(c) shows the LCS before instantiations. Note that a violation of the *GI* constraint from two information instances is shown in red. Fig. 6(d) shows the uninstantiated LCS after the resolution, with the red block in Fig. 6(c) replaced by the blue block in Fig. 6(d). However, a violation of the *LR* constraint is still present in Fig. 6(e) due to the information instance shown in red. Fig. 6(f) shows the potential solution after resolving all violations. Fig. 6(g) shows a possible coalition solution.

### *F. Forming Executable Coalitions*

For searching and setting up coalitions, we assume that all robots are always within communication range, such that every robot can communicate with any other robot when necessary. Given an MS to activate, the robot first creates a solution space and searches through this space for potential coalitions. A coalition is feasible for execution when all the required coalitions are feasible. Note that no coalitions are set up in this phase. Among all feasible coalitions that are found so far (after the given search time has elapsed), IQ-ASyMTRe chooses the one with the least coalition cost to set up. The coalition is only set up when all the required coalitions are also set up. For setting up coalitions in a distributed manner, we use the same request-and-wait negotiation protocol as used in the distributed version of ASyMTRe [34].

### *G. Algorithms and Properties*

The recursive algorithm to generate the solution space is shown in Algorithm 1. It checks all possible schemas to retrieve the required information and recursively checks these schemas. Each reasoning path terminates either in a conflict with the referent instantiation constraint or in a terminal state.

The algorithm to form coalitions in IQ-ASyMTRe is shown in Algorithm 2, which sets up and returns the coalition with the least cost found in a given time. For searching the solution space, the algorithm sequentially checks the potential solutions in a nondescending order that is based on the cost. It activates the required ES–EPS pairs or CSs temporarily to retrieve the information, as specified in the potential solutions. The algorithm then uses the collected information to dynamically determine the feasibility of the coalitions for execution.

Next, we prove important properties of IQ-ASyMTRe, which include the soundness and completeness of the approach.

*Theorem 4.5:* If the set of RPSs expresses all valid information conversions, given sufficient search time, the IQ-ASyMTRe algorithm is complete.

*Proof:* If the set of RPSs expresses all valid information conversions for a given application, then since Algorithm 1 explores all possible ways to retrieve information in a recursive manner, all potential solutions that satisfy the constraints in Lemma 4.4 will be checked. Given that these constraints do not influence the completeness of the solution space, the IQ-ASyMTRe algorithm is complete, given sufficient search time. ■

*Theorem 4.6:* The IQ-ASyMTRe algorithm is sound.

*Proof:* First, since schema connections are made only when the inputs and outputs have matching information types, given that the referent instantiation constraints are respected, solution spaces are created using only valid connections. This also holds for potential solutions. Moreover, since connections are made

until the leaf nodes, which are information sources, the required constant information flow can be maintained. For potential solutions, since one (and only one) branch remains at each *tOR* node, all inputs of the downstream nodes are still satisfied. Hence, all necessary connections are present in the potential solutions to maintain the information flow. Thus, the IQ-ASyMTRe algorithm is sound. ■

### H. Complexity Analysis

The computational and space complexity of Algorithm 1 is equal to the space required to represent the solution space (using an *and–or* tree). To retrieve a single information instance, given the constraints in Lemma 4.4, it is not difficult to conclude that the worst-case complexity for this representation[12] is $O(N_t 2^{N_r} {N_c}^{N_t 2^{N_r}})$, given the following:

1) $N_c$: the maximum number of RPSs producing the same information type;
2) $N_t$: the number of information types related to the information instance to be retrieved;
3) $N_r$: the maximum number of referents associated with information instances for all related information types.

Given a task domain, these numbers are fixed. Hence, the size of the solution space for any information instance is bounded by a constant ${N_c}^{{N_c}^{N_t 2^{N_r}}}$. Meanwhile, the size grows exponentially with the number of information instances to be retrieved. However, most practical problems are still small enough to be computed in reasonable time in practice. For applications in which the size of the solution space is large, the methods in [35] can be applied to achieve online performance by significantly reducing the size of the search spaces for certain problem instances.

In Algorithm 2, the communication complexity for sending information requests to search the entire solution space is bounded by the number of distinct information instances in the solution space, which is bounded by the maximum length of any branching path (i.e., $N_t 2^{N_r}$). However, the complexity is significantly influenced by how the referents of the information instances in the requests are instantiated. If the information instances do not have any instantiated referents, the complexity is only linear in $N_t$, which is based on the *Distinct Requests* constraint. For example, if we have sent a request for $F_G(X)$, no requests need to be sent for $F_G(r_1)$ or $F_G(r_2)$. As the coalitions are setting up, the communication in the distributed system drops gradually until becoming stable, since only a constant information flow is required to maintain the coalitions.

## V. Simulations and Experimental Results

In this section, we provide simulations and experimental results to demonstrate the capabilities of IQ-ASyMTRe for various applications. All simulations are implemented in Player/Stage [36] and are run on a 2.4-GHz Core 2 Duo laptop with 2 GB memory; wall-clock times are reported.

[12]The maximum length of any branching path, subject to all constraints, is $N_t 2^{N_r}$; the number of all possible distinct branching paths is ${N_c}^{N_t 2^{N_r}}$.

TABLE II
Information Required in the Navigation Task

| | |
|---|---|
| $F_G(local)$ | Global position information of the local robot |
| $F_G(goal)$ | Global position information of the goal |
| $F_A(local)$ | Range information (i.e., for obstacle avoidance) |
| $F_M$ | Global map information (i.e., for path planning) |

TABLE III
Potential Solutions of the Navigation Task

| **Potential Solutions for $F_G(local)$** | **Cost** |
|---|---|
| 1. EPS:$F_G(local)$ | 5 |
| 2. CS:$F_G(X)$, EPS:$F_G(X, local)$ | 8 |
| 3. CS:$F_G(X)$, CS:$F_G(local, X)$ | 8.5 |
| 4. CS:$F_G(X)$, CS:$F_G(X, local)$ | 9 |
| 5. CS:$F_G(X)$, CS:$F_R(local, X)$ | 9.5 |
| 6. CS:$F_G(X)$, EPS:$F_R(Z, local)$, CS:$F_R(X, Z)$ | 10.5 |
| 7. CS:$F_G(X)$, CS:$F_R(local, Z)$, CS:$F_R(X, Z)$ | 11.0 |
| 8. CS:$F_G(X)$, CS:$F_R(local, Z)$, CS:$F_R(X, Z)$ | 11.5 |
| 9. CS:$F_G(X)$, CS:$F_R(Z, local)$, CS:$F_R(X, Z)$ | 11.5 |
| 10. CS:$F_G(X)$, CS:$F_R(local, Z)$, CS:$F_R(X, Z)$ | 12.0 |
| 11. CS:$F_G(X)$, CS:$F_R(local, Z)$, CS:$F_R(X, Z)$ | 12.0 |

### A. Simulations

*1) Solution Space and Potential Solutions:* First, we show the solution spaces that are produced by IQ-ASyMTRe for the navigation task, in which the goal is to activate an MS for navigation on different robots. The information instances that are required are listed in Table II along with brief descriptions. The potential solutions are ordered based on the costs of schemas they use[13], which are currently statically defined as follows: cost(RPS) = 0.5, cost(ES–EPS) = 1.0, cost(CS) = 2.0, and cost(MS) = 4.0. $F_G$(goal), and $F_M$ are provided *a priori*, which incur no costs. Since $F_A$(local) can only be retrieved using the local laser sensor, it is also ignored for conciseness. Table III lists the potential solutions for robots that are equipped with a fiducial, a laser, and a GPS sensor; RPSs are not shown for brevity. As an example, the first potential solution involves an ES–EPS node inputting into the required MS, with a total cost of $1 + 4$. Encoding entity information into information enables IQ-ASyMTRe to find solutions (i.e., 6–11) that are not discernible by architectures that use only information types. Hence, more flexibility and completeness are achieved. Note that potential solutions that require the same set of information instances with higher costs can be removed to reduce the search space without affecting the completeness (e.g., using a post-processing algorithm).

*2) Forming Executable Coalitions:* In this simulation, we present a scenario of the navigation task to demonstrate the intuition behind forming executable coalitions. We assume that there are three robots (of type 1 in Table IV) without a localization capability that need to navigate to certain goal positions,

[13]Since the required schemas are always activated during task execution, the cost of a potential solution is proportional to the execution cost. This observation is useful for task allocation.

TABLE IV
INFORMATION USED BY PREVIOUS APPROACHES

| Type | Capabilities | Count |
|---|---|---|
| 1 | Fiducial, Laser, Motor | 3 |
| 2 | Fiducial, Laser, Motor, Localization | 6 |

TABLE V
COALITIONS WITH TWO ROBOTS IN FIG. 7

| Robot | No. Possible Coalitions | No. Executable Coalitions |
|---|---|---|
| $r_R$ | 6 | 2 (with cost 16 and 17.5) |
| $r_G$ | 6 | 1 (with cost 16) |
| $r_Y$ | 6 | 2 (both with cost 16) |

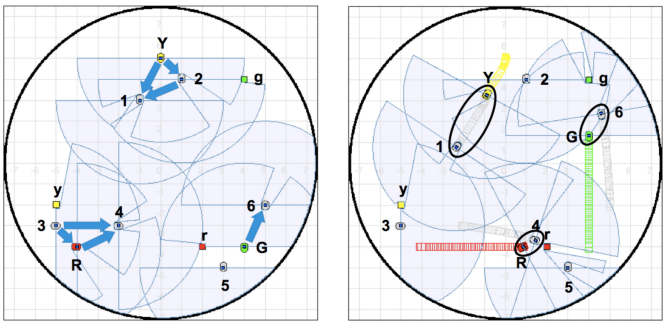



Fig. 7. Forming executable coalitions in the robot navigation task. (a) IQ-ASyMTRe considers the satisfaction of the relative configurations. (b) Formed coalitions are feasible and can be executed.

and there are six other robots of type 2 that can localize. Fiducial sensors on all robots can be used to retrieve relative position information between robots. Previous approaches for forming coalitions use only the information in Table IV, for which any type 2 robots would be considered equivalently by type 1 robots to form coalitions, since there is no way to distinguish between them.

The use of laser or localization sensors does not introduce constraints, since the retrieved information instances $F_G$ are associated with only one referent. However, capability sharing introduces a constraint, since robots must be in the proper physical configuration for one of them to retrieve the relative robot position using a fiducial sensor. Previous approaches do not consider this constraint and thus cannot form executable coalitions. IQ-ASyMTRe considers this constraint by using available information to dynamically instantiate the potential solutions (shown in Table III in this case). For any robot $X$ that can provide $F_G(X)$, the robots also check to see if $F_R(X, \text{local})$ or $F_R(\text{local}, X)$ is retrievable.

For example, when the robots happen to be grouped into three clusters that are relatively distant from each other [but still within the communication range as shown in Fig. 7(a)], it is clear that the consideration of the relative positions is important. The three type 1 robots are shown in red, green, and yellow in Fig. 7(a) (labeled "$R$," "$G$," "$Y$") with the goals shown as small square beacons in their respective colors (labeled "$r$," "$g$," "$y$"). Type 2 robots are labeled from 1 to 6. The FOVs of the fiducial sensors are restricted to be $180^\circ$ facing forward with a maximum range of 4 m (shown as semicircles). When a robot can see another robot, we draw an arrow between the two. Table V summarizes this scenario for coalitions, along with the overall costs for two robots (see Figs. 5 and 12 for how the costs are computed[14]). Fig. 7(b) shows the coalitions that are set up, indicated by ellipses. This illustrates that IQ-ASyMTRe considers the current configuration of the robots when forming coalitions.

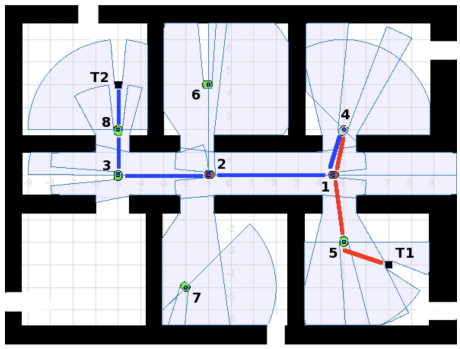
Note that the two potential coalitions for the red robot ("$R$") correspond to potential solutions 2 and 3 in Table III. The coalition with a lower cost (i.e., with the red robot in the back) is preferred. For the green robot ("$G$"), only one executable coalition is found, although the coalition with robot 5 is close to an executable state and may be beneficial when no executable coalitions are found. In such cases, the proximity information can be used by a task planner to create an executable plan for the task. However, discussions about such scenarios are outside the scope of this paper.

*3) Dynamic Monitoring Task:* Next, we show in simulation a scenario of a monitoring task involving more complicated interactions in complex environments, in which the topology of a sensor-connected network[15] of robots is dynamic. In this task, eight mobile robots have to monitor the environment, while keeping a sensor-connected network with fiducial sensors. When targets enter the environment [shown as black squares in Fig. 8(a)], the robots have to provide global positions of the targets. The robots in this simulation are heterogeneous, having different capabilities. The fiducial sensors on robots 1 and 2 have longer ranges with $360^\circ$ FOVs. Only robot 4 can localize. For exploration, the robots simply use the fiducial sensor to search the space while maintaining a sensor-connected network. Fig. 8(b) shows a scenario of the current sensor-connected network of the robots (i.e., with network edges shown as red (narrower) arrows) and two targets ($T_1$ and $T_2$) that enter the environment. In order to provide the target positions, two coalitions are dynamically formed. Members in each coalition and the respective target are sensor-connected, as shown by different line segments in Fig. 8(a); their interactions are shown in Fig. 8(b) as blue (wider) arrows, which are labeled with the communicated information.

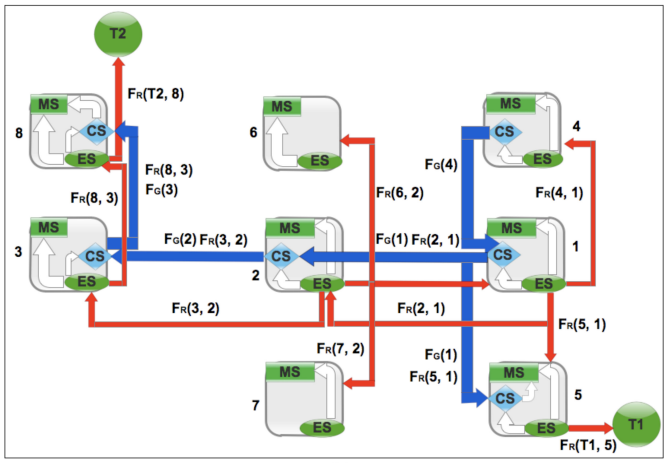
We can see from this simulation that IQ-ASyMTRe can form coalitions with complicated interactions between the robots. The completeness of the solution space guarantees that a solution, when it exists, will be found given sufficient time. When the interactions must be determined dynamically based on the robot capabilities and the current configuration of the robots and the environment, it becomes impractical to always design

[14]Note that how $F_A$ is retrieved is not shown in Fig. 5. In addition to the costs of the coalition solutions shown, there is also a cost to activate a necessary MS (to maintain the relative position) on the robot that is chosen to help, which is the same (i.e., 6) in this simulation. Finally, note that each pair of CSs connected by arrows (between the robots) in the figures is considered as one CS usage.

[15]Two robots are sensor-connected if one robot is in the other's sensor FOV.




(a)




(b)

Fig. 8. Scenario for the dynamic monitoring task with mobile robots. (a) Configuration when targets enter the environment. (b) Connections of the network and the interactions among the robots and the environment.

application-specific methods; thus, general techniques such as that provided by IQ-ASyMTRe can be powerful. Another aspect is that the same robots can appear in different coalitions (such as robots 1 and 4 in this simulation), such that synergy between coalitions can be achieved.

*4) Distributed Search of the Solution Space:* In this simulation, we analyze how fast IQ-ASyMTRe finds solutions distributively in difficult-to-search scenarios. We first provide a formulation of the search time in general scenarios. The search process always starts with the local robot $r_L$ (or denoted by $r^{(0)}$). We denote $R^{(1)}$ as the set of robots from which $r_L$ requests help. For any robot $r^{(1)} \in R^{(1)}$, it may recursively request information from another set of robots, which is denoted by $R^{(2)}$; any $r^{(2)} \in R^{(2)}$ can recursively request information, and the process continues. Assuming that the maximum step of this recursive process for the search is $N$ and there is no message loss, the time required to find the coalition is bounded by

$$T = \max_{r^{(0:N)}} \left( \sum_{k=0}^{N-1} T_D\left(r^{(k)}, r^{(k+1)}\right) + T_D\left(r^{(k+1)}, r^{(k)}\right) + \sum_{k=0}^{N} P\left(r^{(k)}\right) \cdot S\left(r^{(k)}\right) \cdot T_G \right) + 2 \cdot N \cdot T_C \qquad (1)$$

in which $r^{(0:N)}$ are robots in a recursive path, $P(r^{(k)})$ represents the index of the potential solution (after ordering) used by $r^{(k)}$, and $S(r^{(k)})$ represents the number of solution spaces that are created by $r^{(k)}$ during the search. $T_G$ is a constant time gap inserted between checking two consecutive potential solutions; the message queue sends received messages to process every $T_C$ seconds. $T_G$ and $T_C$ are introduced due to a thread implementation. $T_D(r_1, r_2)$ is the communication delay to send (from $r_1$) and receive a message (by $r_2$).

For a concrete example, we create a scenario in which all robots are aligned in a column formation, and there is only one robot with a localization capability (i.e., the one in the front). All robots are assumed to be facing the front and have a fiducial to detect position information relative to nearby robots in the front. We are interested in determining how long the robot in the back takes to find a coalition solution. Since the view of any robot is blocked by the one immediately in front of it, the localization capability of the front-most robot is shared in a sequential order from the nearest to the farthest robot.[16]

Fig. 9 shows the time used by the farthest robot to find the coalition solution (which must include all robots) as the number of robots increases, averaged over five runs. As $N$ (i.e., the number of robots without a localization capability) in (1) increases, the number of hops that the information must travel also increases. Although the joint search space is exponential in the number of robots, we can see that the algorithm of IQ-ASyMTRe is able to distributively find the coalition involving ten robots ($N = 9$) in about 30 s, when $T_G$ and $T_C$ are set to be 0.2 s. In this simulation, $T_D$ is implemented to follow a normal distribution $N(\mu, \sigma)$ in which $\mu = 3.85$ ms and $\sigma = 5.05$ ms. (These communication parameters are based on physical experiments with the Pioneer robots in our laboratory, which use the 802.11n wireless standard.) $P(r^{(k)})$ is bounded by 11 (see Table III) and $S(r^{(k)})$ is bounded by 10, since only global and relative position information (relative to the other nine robots) is involved. We can see that the results in Fig. 9 are consistent with (1).

Fig. 9 also illustrates (in the lower two figures) the messages sent and received for the scenario with ten robots in one of the runs. Peaks in the plots for the sent messages (the figure in the center) are created when robots share the retrieved information instances with others. The farthest robot finds the coalition solution at the 28th second, which corresponds to the black peak (with circle data point) in the figure. We can also see the phase delay of the peaks as we gradually move away from the front-most robot, which reflects the hops of information. Compared with the plots for the sent messages, the plots for the received messages are more in phase. This is due to the fact that information requests are often addressed to all robots (whenever there are uninstantiated referents).

A 60-s gap is manually inserted to clearly separate the searching phase from the maintaining phase (after setting up a coalition). In Fig. 9, we can also see that the numbers of sent or

[16]For example, any robot (except the front-most one) can only provide its localization information to the robot behind it after it localizes itself, which is achieved from the retrieval of its relative position to the robot immediately in front and the robot's global position.

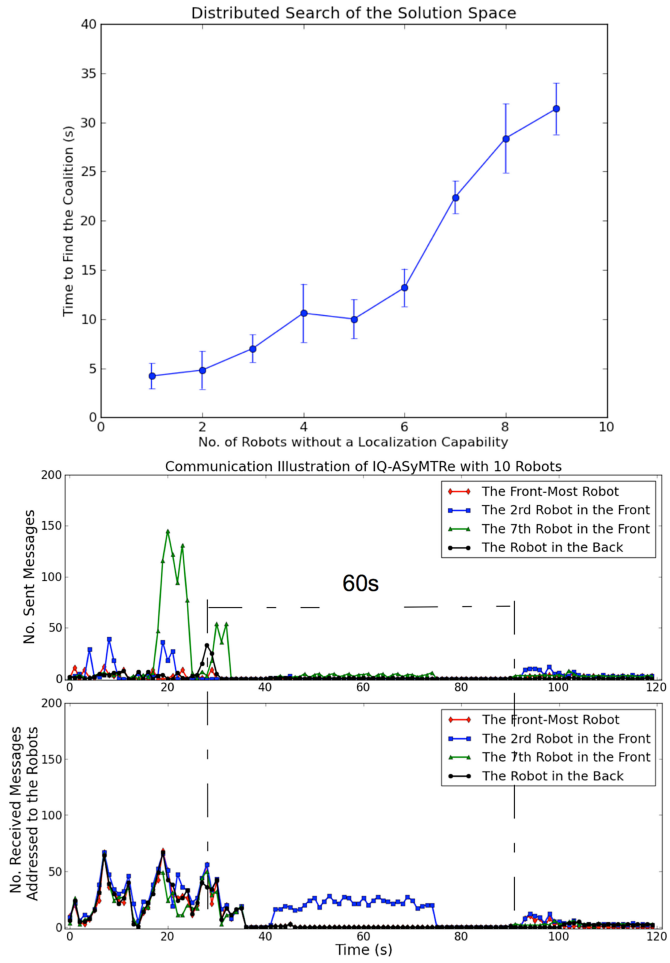



Fig. 9. Distributed search of the solution space in difficult-to-search scenarios. The top figure shows the time to find the coalition solution as the number of robots increases. The bottom two figures show the messages sent and received (that are addressed to individual robots) for four of the robots in the 10-robot ($N = 9$) scenario.

received messages become stable after the coalition is set up (after the 90th second), which represents the constant information flows (between different robots) required to maintain the coalitions. We can see from this simulation that IQ-ASyMTRe can quickly find solutions even for scenarios that are difficult to search in distributed systems.

*5) Cooperative Robot Box Pushing Task:* Next, we show how executable solutions can be used to solve a cooperative robot box pushing task in a general scenario. Instead of pushing in a given direction as discussed in [32], the robots are more often required to push boxes to specified locations. For oversized boxes, it is more efficient for robots to push cooperatively, since the pushing direction of the box may need to be realigned frequently through rotations. The solution space for this task with two robots is presented in Fig. 10, in which the goal position is known. The bumper information is shared, and used to determine whether the teammate is ready to push, to synchronize their behaviors.

The initial configuration is shown in Fig. 11(a), in which the purple box (labeled "1") needs to be pushed to the position

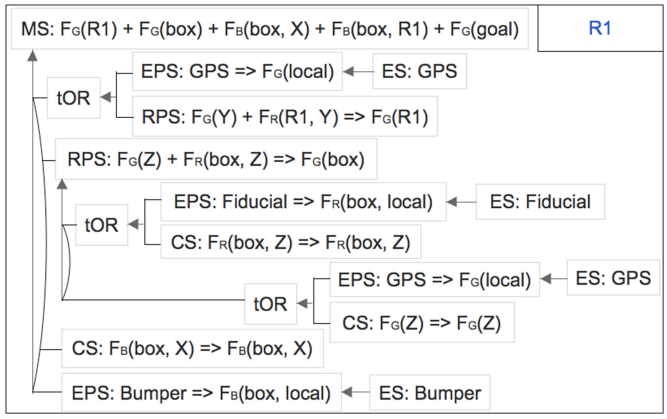



Fig. 10. Solution space that encodes the potential solutions used in the cooperative robot box pushing task in simulation.

of the blue box (labeled "2"). There are five robots of three different types in the environment. Robots from the first type (labeled "*R*") are equipped with a bumper sensor to push the box; robots from the second type (labeled "*L*") are equipped with a GPS sensor for localization; the last type (labeled "*G*") has both sensors. All robots are also equipped with a fiducial and a laser sensor. Since box "2" is in the way of box "1," another MS for box pushing (with a single robot) must be activated on a robot to push "2" out of the way.[17] As a result, three robots with a bumper sensor need to be assigned (in this case, two "*R*"s and one "*G*" are assigned). However, directly executing the MSs for box pushing in the initial configuration is not possible due to the unavailability of the required information $F_G(\text{box})$. To obtain the missing information, three MSs for navigation must be activated first on these three robots to find the boxes. Since two of the three robots cannot localize (two "*R*"s), they set up a coalition with two robots that can (two "*L*"s), respectively. Once the boxes are found, the MSs for box pushing can be activated. Fig. 11 shows snapshots from this simulation. A supplemental video of this simulation is also attached.

To use IQ-ASyMTRe for different tasks, one only needs to implement the behaviors (MSs) and specify the required information. IQ-ASyMTRe is then able to reason about possible ways to retrieve the information in the current situation. In cases when certain information is not retrievable, higher level task planning may be required, which is manually implemented in this simulation. The first step toward creating task plans autonomously based on the missing information identified by IQ-ASyMTRe is presented in [38].

## B. Physical Experiments

*1) Cooperative Robot Navigation Task:* To demonstrate the flexibility of IQ-ASyMTRe, we create two scenarios for the navigation task with physical robots, as shown in Figs. 13 and 14. Instead of working with fiducial sensors, the robots are equipped

[17] Since these two tasks are dependent, they may require sequential execution (using the approach in [37]) to avoid conflicts, such as one box being pushed into other robots or boxes. In the attached video for this simulation, these two tasks are executed simultaneously to shorten the running time.

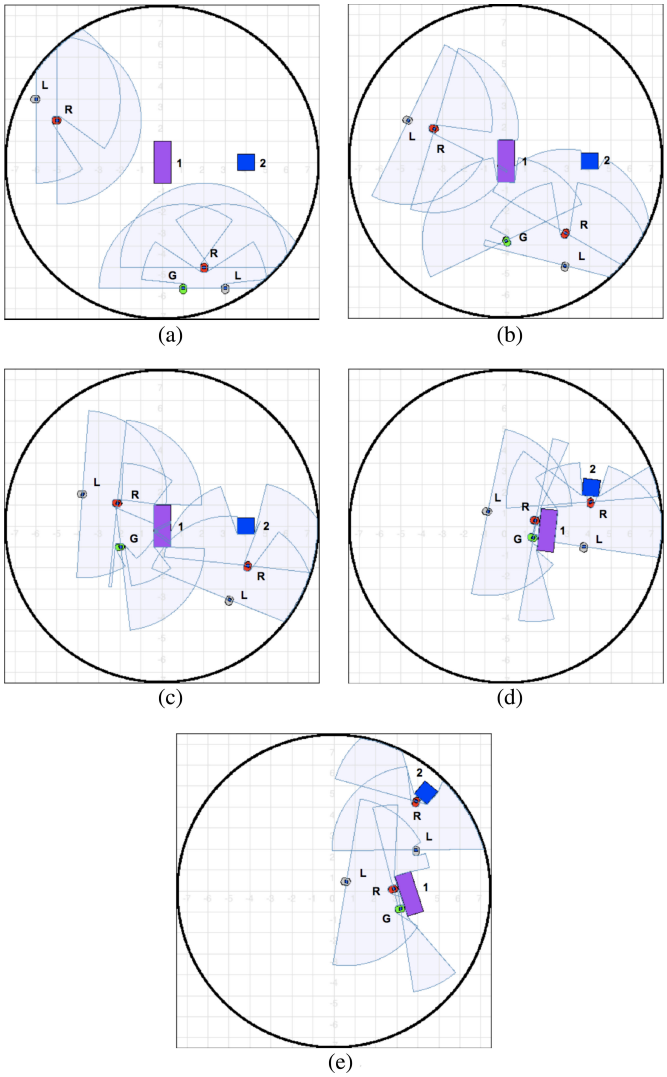



Fig. 11. Robots performing a cooperative box pushing task in a general scenario, in which the goal is to push the purple box ("1") to the blue box ("2"). (a) Initial configuration in which robots are distributed in two distant clusters. Two robots with a bumper need to be assigned to activate the MS for cooperative box pushing, while one is needed to push box "2" out of the way. (b) Robots ("$L$") with a localization capability help two of the robots assigned for box pushing to localize to find the boxes. (c) Robots are in positions ready to push. (d) Robots start pushing the boxes; two of them are being helped to achieve localization via constant information sharing; the bumper information between the two robots assigned for the cooperative box pushing is also constantly shared. (e) Task is completed. (a) 0 s. (b) 40 s. (c) 80 s. (d) 305 s. (e) 635 s.

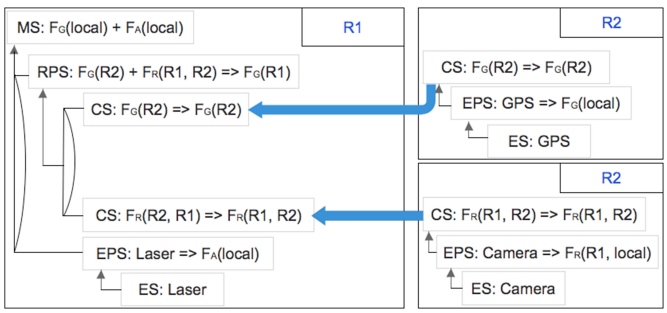



Fig. 12. Coalition solution with the robot that can localize in the back.

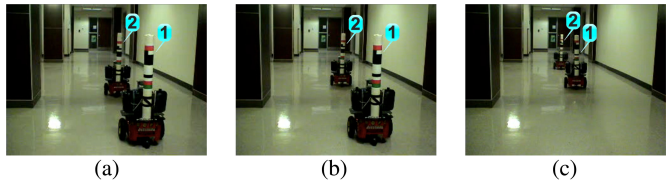



Fig. 13. Robots in a navigation task, with the robot with a localization capability (labeled "2") in front. (a) Initial configuration with robot "2" in the front. (b) Robot "2" goes to the goal, while the other robot without the localization capability (labeled "1") is trying to set up a coalition with it. (c) Coalition is set up, and the robots navigate through the environment.

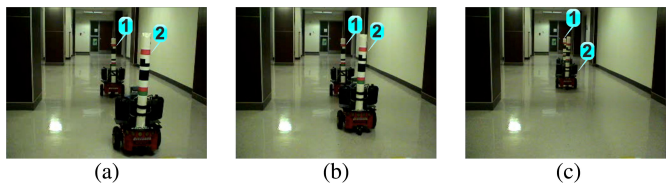



Fig. 14. Robots in a navigation task, with the robot with a localization capability (labeled "2") in back. (a) Initial configuration with robot "2" at the back. (b) Robot "2" starts first while robot "1" is trying to set up a coalition with it. (c) Coalition is set up, and the robots start navigation.

with camera sensors pointing forward (with a 60° FOV) to retrieve the relative positions of nearby robots. The robot with a localization capability (labeled "2") is in front of the robot without it (labeled "1") in one scenario and is behind robot "1" in the other. The coalition solution for the first scenario is similar to that shown in Fig. 5; the coalition solution for the second scenario is shown in Fig. 12. One can compare these solutions with Fig. 2 to see the differences from the ASyMTRe architecture. Both robots are assigned to activate the MS for navigation to go to the same position. For both scenarios, the robot with the localization capability starts execution first since it is self-sufficient for the MS. This experiment shows that the IQ-ASyMTRe architecture can distinguish among different robot configurations in the current situations and form executable coalitions accordingly.

*2) Cooperative Robot Box Pushing Task:* Finally, with physical robots, we show that IQ-ASyMTRe can provide flexible and robust solutions. We illustrate several scenarios in the cooperative box pushing task with different robot capabilities and environmental settings, in which IQ-ASyMTRe provides different solutions. Unlike in the simulations, we assume that the robots can localize, but $F_G$(goal) is not given and has to be retrieved. This is useful for tasks with dynamic goal positions (e.g., pushing a box while following a person). The bumper information and the box's relative position to the robot are approximated using the sonar sensors.

We start with the simplest scenario as shown in Fig. 15(a), in which both robots (referred to as robot $L$ and $R$) assigned to activate the MS can see the goal marker. In Fig. 15(b), we add a view blocker that blocks $R$ from viewing the goal marker. We block both robots from viewing the goal marker in the third scenario, as shown in Fig. 15(c). Meanwhile, we also add an intermediate robot $I$, which can localize itself and can see the goal marker. The last scenario, which is shown in Fig. 15(d), is

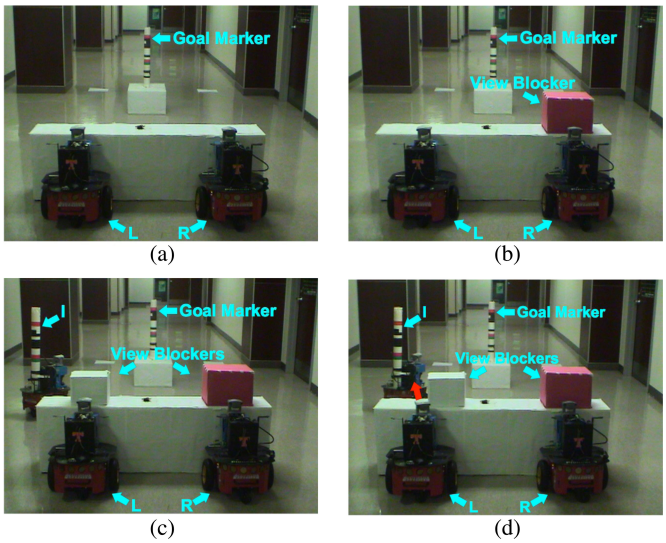



Fig. 15. Cooperative box pushing task. (a) Simple box pushing scenario where the robots assigned to activate the MS can see the goal marker. (b) Right robot's view is blocked. (c) Both robots' views are blocked, while the intermediate robot can localize and can see the goal. (d) Both robots are blocked and the intermediate robot can see the goal but cannot localize. In all scenarios, the barcode markers are used to extract the relative position information using cameras.

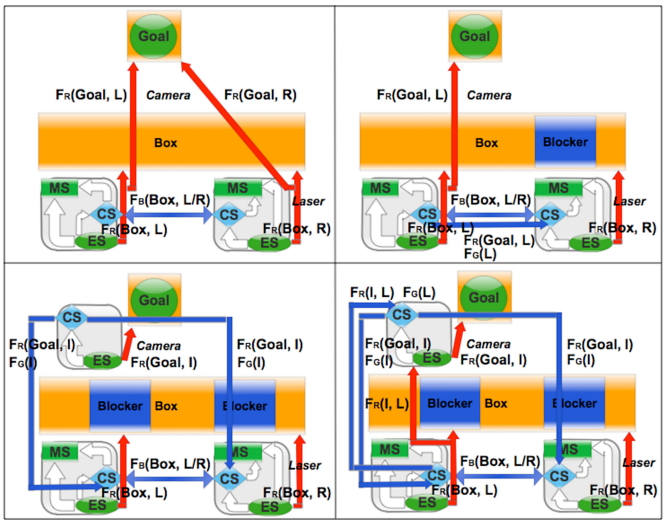



Fig. 16. Information flow for scenarios shown in Fig. 15.

the most interesting case: We remove the localization capability from robot $I$. In this last scenario, robot $L$ obtains $F_G(\text{goal})$ by first helping robot $I$ to localize by providing $F_R(I, L)$ and $F_G(L)$. Robot $I$ then starts helping robot $L$ and $R$ in retrieving $F_G(\text{goal})$. In this scenario, the helper also needs to be helped in order to accomplish the task. Fig. 16 illustrates the various ways that information flows among the robots for these four scenarios.

## VI. Conclusion and Future Work

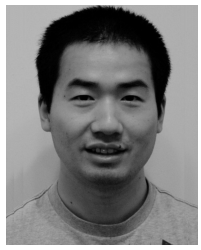
This paper has discussed the IQ-ASyMTRe approach, which forms executable coalitions for multirobot tasks, in which robots can share sensory and computational capabilities. This approach significantly extends the previous ASyMTRe architecture to consider the current robot configuration and environment settings to form coalitions. The soundness and completeness of the approach is proven. We show that IQ-ASyMTRe can dynamically and flexibly form coalitions that are executable for various multirobot tasks. To the best of our knowledge, this is the first attempt to create a general solution to form executable coalitions for multirobot tasks.

In future work, we plan to study techniques to reduce the computational requirements so that scalability can be improved for systems of moderate to large sizes (e.g., 100 robots). Other interesting aspects include addressing sensor fusion [28] across multiple robots.

## References

[1] V. Sujan and S. Dubowsky, "Visually guided cooperative robot actions based on information quality," *Auton. Robots*, vol. 19, no. 1, pp. 89–110, Jul. 2005.
[2] L. E. Parker and F. Tang, "Building multirobot coalitions through automated task solution synthesis," *Proc. IEEE*, vol. 94, no. 7, pp. 1289–1305, Jul. 2006.
[3] P. Shiroma and M. Campos, "CoMutaR: A framework for multi-robot coordination and task allocation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 4817–4824.
[4] Y. Zhang and L. E. Parker, "IQ-ASyMTRe: Synthesizing coalition formation and execution for tightly-coupled multirobot tasks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 5595–5602.
[5] S. Botelho and R. Alami, "M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Proc. IEEE Robot. Autom.*, 1999, vol. 2, pp. 1234–1239.
[6] M. Dias and A. Stentz, "A free market architecture for distributed control of a multirobot system," in *Proc. 6th Int. Conf. Intell. Auton. Syst.*, 2000, pp. 115–122.
[7] C. Fua and S. Ge, "COBOS: Cooperative backoff adaptive scheme for multirobot task allocation," *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1168–1178, Dec. 2005.
[8] B. Gerkey and M. Mataric, "Sold!: Auction methods for multi-robot coordination," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 758–768, Oct. 2001.
[9] B. Werger and M. Mataric, "Broadcast of local eligibility for multi-target observation," in *Proc. 5th Int. Conf. Distrib. Auton. Robot. Syst.*, New York: Springer-Verlag, 2000, pp. 347–356.
[10] R. Zlot and A. Stentz, "Complex task allocation for multiple robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 1515–1522.
[11] R. Zlot, A. Stentz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, vol. 3, pp. 3016–3023.
[12] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multirobot cooperation," *IEEE Trans. Robot. Autom.*, vol. 14, no. 2, pp. 220–240, Apr. 1998.
[13] B. Gerkey and M. Mataric, "MURDOCH: Publish/subscribe task allocation for heterogeneous agents," in *Proc. 4th Int. Conf. Auton. Agents*. New York: ACM Press, 2000, pp. 203–204.
[14] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 813–825, Oct. 2002.
[15] R. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. C-29, no. 12, pp. 1104–1113, Dec. 1980.
[16] T. Lemaire, R. Alami, and S. Lacroix, "A distributed tasks allocation scheme in multi-UAV context," in *Proc. IEEE Int. Conf. Robot. Autom.*, Jul. 2004, vol. 4, pp. 3622–3627.
[17] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," *Artif. Intell.*, vol. 101, no. 1–2, pp. 165–200, 1998.
[18] O. Shehory and T. Kraus, "Feasible formation of coalitions among autonomous agents in non-super-additive environments," *Comput. Intell.*, vol. 15, no. 3, pp. 218–251, 1999.
[19] M. Klusch and A. Gerber, "Dynamic coalition formation among rational agents," *IEEE Intell. Syst.*, vol. 17, no. 3, pp. 42–47, May/Jun. 2002.

[20] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme, "Coalition structure generation with worst case guarantees," *Artif. Intell.*, vol. 111, no. 1–2, pp. 209–238, 1999.
[21] T. Sandholm and V. Lesser, "Coalition formation among bounded rational agents," in *Proc. 14th Int. Joint Conf. Artif. Intell.*, Jan. 1995, pp. 662–669.
[22] D. Lyons and M. Arbib, "A formal model of computation for sensory-based robotics," *IEEE Trans. Robot. Autom.*, vol. 5, no. 3, pp. 280–293, Jun. 1989.
[23] R. Arkin, "Motor schema based navigation for a mobile robot: An approach to programming by behavior," in *Proc. IEEE Int. Conf. Robot. Autom.*, Mar. 1987, vol. 4, pp. 264–271.
[24] R. Fikes and N. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artif. Intell.*, vol. 2, no. 3–4, pp. 189–208, 1971.
[25] A. Saffiotti, "Fuzzy logic in autonomous robotics: Behavior coordination," in *Proc. 6th IEEE Int. Conf. Fuzzy Syst.*, Barcelona, Spain, 1997, pp. 573–578.
[26] T. Estlin, R. Volpe, I. Nesnas, D. Mutz, F. Fisher, B. Engelhardt, and S. Chien, "Decision-making in a robotic architecture for autonomy," in *Proc. Int. Symp. Artif. Intell., Robot., Autom. Space*, 2001, papers 92152–7383.
[27] R. R. Murphy, "Dempster-Shafer theory for sensor fusion in autonomous mobile robots," *IEEE Trans. Robot. Autom.*, vol. 14, no. 2, pp. 197–206, Apr. 1998.
[28] A. Stroupe, M. Martin, and T. Balch, "Distributed sensor fusion for object position estimation by multi-robot systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2001, vol. 2, pp. 1092–1098.
[29] N. Kalra, D. Ferguson, and A. Stentz, "Hoplites: A market-based framework for planned tight coordination in multirobot teams," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 1170–1177.
[30] L. Vig and J. Adams, "Multi-robot coalition formation," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 637–649, Aug. 2006.
[31] J. Spletzer and C. Taylor, "Sensor planning and control in a dynamic environment," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, vol. 1, pp. 676–681.
[32] B. Donald, J. Jennings, and D. Rus, "Information invariants for distributed manipulation," *Int. J. Robot. Res.*, vol. 16, no. 5, pp. 673–702, 1997.
[33] B. Gerkey and M. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, Sep. 2004.
[34] F. Tang and L. E. Parker, "Distributed multi-robot coalitions through ASyMTRe-D," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Aug. 2005, pp. 2606–2613.
[35] Y. Zhang and L. E. Parker, "Solution space reasoning to improve IQ-ASyMTRe in tightly-coupled multirobot tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 370–377.
[36] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc. 11th Int. Conf. Adv. Robot.*, 2003, pp. 317–323.
[37] Y. Zhang and L. E. Parker, "Considering inter-task resource constraints in task allocation," *Auton. Agents Multi-Agent Syst.*, 2012, 1–31. DOI: 10.1007/s10458-012-9196-7.
[38] Y. Zhang and L. E. Parker, "Task allocation with executable coalitions in multirobot tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 3307–3314.



**Yu Zhang** (M'09) received the B.S. degree in software engineering from the Huazhong University of science and Technology, Wuhan, China, in 2006 and the M.S. and Ph.D. degrees in computer science from the University of Tennessee, Knoxville (UTK), in 2009 and 2012, respectively.

He joined the Department of Electrical Engineering and Computer Science, UTK, as a Graduate Student in August 2007. He has been conducting research with the Distributed Intelligence Laboratory, UTK, since 2008, where he has been involved with a project funded by the National Science Foundation. His current research interests include distributed robot systems, mainly investigating how heterogeneous robots can reason about forming coalitions based on the current robot team configurations and environment settings and, then, how robot coalitions can autonomously and flexibly execute the assigned tasks. His research interests also include multiagent systems, planning algorithms, machine learning, and sensor fusion.

**Lynne E. Parker** (F'10) received the B.S. degree from Tennessee Technological University, Cookeville, in 1983 and the M.S. degree from the University of Tennessee, Knoxville, in 1988, both in computer science, and the Ph.D. degree in computer science, with a minor in brain and cognitive science, from the Massachusetts Institute of Technology (MIT), Cambridge, in 1994.

At MIT, she conducted research on cooperative control algorithms for heterogeneous multirobot systems with the Artificial Intelligence Laboratory. She is a Professor and an Associate Head with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, where she serves as the Director of the Center for Intelligent Systems and Machine Learning, as well as the Distributed Intelligence Laboratory. She also holds an appointment as an Adjunct Distinguished Research and Development Staff Member at Oak Ridge National Laboratory, Oak Ridge, TN, where she was a Full-Time Researcher for several years. Her research interests include distributed robotics, heterogeneous teams, human–robot interaction, sensor networks, and machine learning.

Dr. Parker is a Former Editor and Associate Editor of IEEE TRANSACTIONS ON ROBOTICS.