Proc. of IEEE International Conference on Robotics and Automation, Shanghai, China, 2011.

Solution Space Reasoning to Improve IQ-ASyMTRe in Tightly-Coupled Multirobot Tasks

Yu Zhang and Lynne E. Parker

***Abstract*— In our prior work, we proposed the IQ-ASyMTRe architecture as a general method to combine coalition formation and execution for tightly-coupled multirobot tasks. IQ-ASyMTRe extends the ASyMTRe architecture by introducing several new mechanisms to provide more flexibility for coalition formation as well as to facilitate coalition execution. On the other hand, these mechanisms also change the process of reasoning about solutions and further increase the complexity of the solution space. In this paper, we provide improvements for utilizing the IQ-ASyMTRe architecture based on reasoning about the solution space. We introduce a method in which the exponential growth of the number of potential solutions to be searched can be avoided; instead, the search space is only of linear size for certain tasks. Unnecessary potential solutions are removed to further increase online efficiency. Moreover, the relationships between the created solution space and the complete solution space are studied, and are utilized to provide more coverage of the complete solution space for arbitrary tasks. Although these improvements are discussed with respect to IQ-ASyMTRe, they are also applicable to architectures that approach the generality that IQ-ASyMTRe achieves. Robot simulation and experimental results are provided to demonstrate that the generation and searching of the solution space can be done online (which was impractical previously even for tasks with relatively modest complexities) for certain tasks, and to illustrate how our approach impacts the solution space.**

I. INTRODUCTION

Many architectures [3], [5], [7] are proposed which aim at solving the coalition formation problem with multi-robot teams, in which each assigned task may require the cooperation of multiple robots, as individual robots may not have all the capabilities to accomplish the task. The coalition formation problem has been shown to be NP-hard and the tasks are often referred to as multirobot tasks [4]. In order to reason about robot coalitions to solve multirobot tasks, while most of the architectures divide these tasks into subtasks or roles that individual robots can accomplish, the ASyMTRe [6] (Automated Synthesis of Multi-Robot Task Solutions through Software Reconfiguration) architecture enables a finer resource sharing by defining robot capabilities in terms of sensory and computational level schemas, as well as communication and motor schemas. As a result, ASyMTRe is able to reason about how to accomplish multirobot tasks through utilizing these schemas, hence providing a more flexible method for forming coalitions. However, an issue with ASyMTRe is that robot configurations and dynamic and environmental factors are not considered, hence limiting the application of the architecture to arbitrary tasks.

In our previous work [9], we introduce the IQ-ASyMTRe (extended ASyMTRe with Information Quality) architecture, which extends the ASyMTRe architecture and utilizes information quality measures [8] to account for robot configurations and dynamic and environmental factors, so as to provide a method for combining coalition formation and execution in the same architecture. These extensions significantly change the solution space created by the reasoning process. While IQ-ASyMTRe guarantees forming executable coalitions and provides even more flexibility, the computational complexity for reasoning about coalition solutions also increases. The complexity needs to be greatly reduced to facilitate online execution. Moreover, the relationships between the created solution space and the complete solution space need to be studied. Such relationships can be utilized, when necessary, to provide more coverage of the complete solution space so as to further improve the flexibility and robustness of the approach.

In this paper, we formally study the influence of these extensions on the reasoning of the solution space for the IQ-ASyMTRe architecture and provide improving techniques. A review of the IQ-ASyMTRe architecture is first given to provide the basic background knowledge (Section II), which includes formal definitions of information type and information conversion. In Section III, after studying the complexity of the solution space, we introduce a way to restrict the exponential growth of the number of potential solutions required to be searched for tasks with certain properties, such that the online generation and searching of the solution space becomes practical. Then, we discuss how unnecessary potential solutions can be efficiently removed from the solution space to further increase online efficiency. Finally, we continue our discussion on the relationships between the created solution space and the complete solution space for arbitrary tasks. Although full coverage of the complete solution space cannot be guaranteed, an approximation algorithm is provided. Simulation and experimental results are provided in Section IV that illustrate the importance of these improving techniques. Finally, we make some conclusions in Section V.

II. THE IQ-ASYMTRE ARCHITECTURE

The ASyMTRe architecture [6] defines basic building blocks of robot capabilities to be collections of environmental sensors (ESs), perceptual schemas (PSs), motor schemas (MSs) [1], and communication schemas (CSs). Each schema

This material is based upon work supported by the National Science Foundation under Grant No. 0812117.

Yu Zhang and Lynne Parker are with the Distributed Intelligence Laboratory in the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996-3450, USA, {yzhang51,parker}@eecs.utk.edu.

can be activated when its input is satisfied and may produce certain output. To characterize the input and output, a set of information types $F = \{F_1, F_2, ...\}$ is introduced to label them. In ASyMTRe, information types differ from data types (e.g., integers) in that they have semantic meanings (e.g., a robot's global position). Then, according to a set of rules, connections can be created among the schemas on the robots to allow information to flow through the system to activate the required motor schemas to accomplish the tasks.

However, the inconsideration of robot configurations and dynamic and environmental factors of ASyMTRe prevents its application to arbitrary tasks. First of all, as the semantic meanings of the information are not fully captured in the definition of information type, potential solutions do not always represent feasible solutions. Another problem with ASyMTRe is that information conversion is not explicitly modeled. Furthermore, dynamic and environmental influence is not considered. To address the above issues, we first introduce the extended architecture, IQ-ASyMTRe, in our previous work [9]. IQ-ASyMTRe introduces a complete definition of information type and explicitly models information conversion by using a special type of perceptual schemas. To account for dynamic and environmental influence, IQ-ASyMTRe incorporates information quality measures [8]. We provide formal definitions of the extensions of representation as follows.

A. Information Type and Information Instance

The incompleteness of the definition of information type in ASyMTRe is due to the fact that the relationships between entities (which can be locations, agents or other objects that can be identified in the environment) and information are not specifically captured. Intuitively, information must be specified with a set of referents. For example, the information of r_A*'s global position* is meaningless without specifying r_A. IQ-ASyMTRe uses both *information type* and *information instance* for a complete specification of information. We use $\mathcal{F}$ to represent the information type in ASyMTRe.

Definition 2.1: Information Type – An information type in IQ-ASyMTRe is specified by a pair, ($\mathcal{F}$, N), where N is the number of referents that should be associated with $\mathcal{F}$.

In order to reason about the complete semantic meaning of information, information type alone is not sufficient.

Definition 2.2: Information Instance – An information instance of a particular information type, ($\mathcal{F}$, N), can be represented as $F(Ref_{1:N})$, where Ref_i is used to refer to the ith referent for the information instance.

Each referent, Ref_j, can be instantiated to a particular entity or remain uninstantiated, waiting for a 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 that X is instantiated to.

B. Information Conversion (Rule PS)

The Rule PS (denoted by *RPS* henceforth) is introduced in IQ-ASyMTRe as a special type of PS to express information conversions. RPSs expressed in IQ-ASyMTRe can be specified in the Backus-Naur Form (BNF) as follows.

TABLE I
RPS'S USED IN OUR APPROACH

RPS	Description
1. $F_G(X)$+$F_R(Y,X) \Rightarrow F_G(Y)$	global+relative $\Rightarrow$ global
2. $F_R(Y,X) \Rightarrow F_R(X,Y)$	relative $\Rightarrow$ relative
3. $F_R(X,Z)$+$F_R(Y,Z) \Rightarrow F_R(X,Y)$	relative+relative $\Rightarrow$ relative
4. $F_G(X)$+$F_G(Y) \Rightarrow F_R(Y,X)$	global+global $\Rightarrow$ relative

Definition 2.3: Information Conversion – Information conversions in IQ-ASyMTRe express relationships between composite information instances (abbreviated as comp ins in the BNF specifications).

The composite information instance on the right hand side can be converted from the composite information instance on the left hand side using the information conversion:

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

The BNF of a composite information instance is given as follows, in which *info ins* is an abbreviation for *information instance*. Information instances are combined into composite information instances using the operators {*iAND*, *iOR*}, which have similar meanings as the *AND* and *OR* operators in propositional logic. Since *iOR* operators on the right hand side are not well defined, the definition for the two are different:

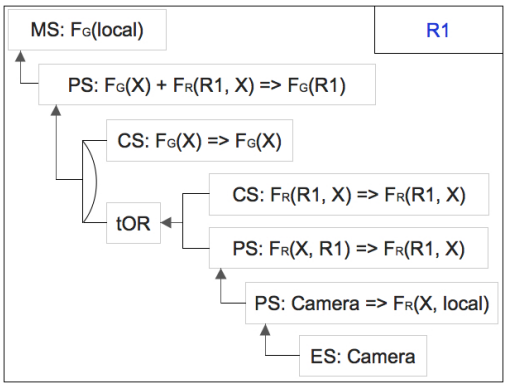
$$\begin{aligned} < \textit{l-comp ins} > ::= & (< \textit{l-comp ins} > \textit{iAND} < \textit{l-comp ins} >) \\ & \mid (< \textit{l-comp ins} > \textit{iOR} < \textit{l-comp ins} >) \\ & \mid < \textit{info ins} > \end{aligned}$$

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

For the following discussions, we denote the *iAND* operator by '+', for conciseness. Table I shows some basic RPSs. The conversions are general since the referents can be instantiated to different entities. However, constraints exist such that the same referent labels must be instantiated to the same entities.

C. Solution Space and Potential Solutions

Note that both the ASyMTRe and IQ-ASyMTRe are inspired by the information invariants theory [2]. Hence, we can consider schemas as *components*. The solution reasoning process then is to check how these *components* can be *permuted* in various situations (e.g., with different robot *configurations*) to build *sensori-computational systems* providing the same functionalities. These systems are hence *equivalent* for the current task. Based on this understanding, potential solutions represent ways to connect these *components* within the systems. The difference, though, is that the reasoning process of ASyMTRe and IQ-ASyMTRe is concentrated on the semantic meanings extracted from the input and



(a)

Fig. 1. (a) The graphical representation of a solution space for a robot to obtain its global position with only a camera sensor. The referent *local* refers to the robot itself. The solution space encodes two potential solutions. One is to have another robot send over its global position (CS: $F_G(X) \Rightarrow F_G(X)$) and use the camera sensor to sense the relative position of the robot to itself (PS: Camera $\Rightarrow F_R(X, local)$). 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 another robot.

output of *components* and hence avoids much mathematical complexity.

A solution space encodes all potential solutions for retrieving certain information instances needed to solve a given multirobot task. In order to create the solution space, the IQ-ASyMTRe reasoning algorithm checks all schemas that can output the input information instances to activate the required motor schema. The algorithm then checks recursively for the inputs of those schemas. Figure 1(a) shows a solution space for retrieving the global position information for a motor schema. The *tOR* node is introduced to manage multiple options of connection. In order to simplify the solution space, additional constraints for schema connections are introduced in IQ-ASyMTRe. (For details, see [9].) After the solution space is created, potential solutions can be extracted from the solution space by making decisions on which schema node to use at each *tOR* node (the rest of the nodes are trimmed), as the extraction proceeds from the root to the leaves.

III. IMPROVEMENTS FOR IQ-ASYMTRE

In the previous section, we have provided some basic knowledge of IQ-ASyMTRe and discussed how solution space is created. Next, starting with analyzing the solution space, we discuss issues with IQ-ASyMTRe and provide improving techniques.

A. Complexity of the Solution Space

As we can see from the previous section, the solution space is created based on the required input information instances for the motor schemas. For multi-robot systems, the size of the complete solution space is clearly exponential. In this respect, IQ-ASyMTRe aims to reduce the size of the solution space by allowing uninstantiated referents and restricting schema connections. We first analyze their impact on the solution spaces with a single required input information instance about which to reason. Figure 1(a) gives an example of such a solution space. The following notations are defined for the complexity analysis:

- N_c: the maximum number of information conversions producing the same information type.
- N_t: the number of information types related to (i.e., through RPSs) the information instance to be reasoned.
- N_r: the maximum number of referents associated with information instances for all related information types.

Since the maximum length of any reasoning path, subject to all constraints imposed in IQ-ASyMTRe, is $O(N_t 2^{N_r})$, the number of all distinct reasoning paths is then $O({N_c}^{N_t 2^{N_r}})$. Since all reasoning paths are distinct in IQ-ASyMTRe according to the reasoning process, the worst case complexity of both time and space for the solution space is $O(N_t 2^{N_r} {N_c}^{N_t 2^{N_r}})$. Notice that all notations in this complexity analysis (i.e., N_c, N_t, N_r) represent constants with respect to certain problem domains and are independent of variables such as the number of robots. Hence, the complexity of the solution space for reasoning about a single information instance is constant given the problem domain, as is the number of potential solutions in the solution space.

Now, let us extend the discussions to the more complex cases and suppose that the number of required information instances for activating a desired motor schema is N_i. Although the space and time complexity for the creation of the solution space is linear with respect to N_i, the number of potential solutions grows exponentially. Fortunately, very much like the fact that independencies among random variables can restrict the exponential growth of the joint probability table, we can define similar *independence* relationships between information instances to address this problem.

Definition 3.1: Independence of Information Instance – An information instance is independent of another if there are no uninstantiated referents labeled the same in both information instances (such referents are required to be instantiated to the same entity in IQ-ASyMTRe).

In other words, an information instance is independent of another if there are no constraints for referent instantiation between the two, since then the two information instances can be reasoned about separately without interfering with each other. The notion of independence of information instance can be easily extended to sets of information instances, such that the independence relationships divide the information instances into mutually independent sets and the number of potential solutions that must be searched can be significantly reduced. If the maximum number of information instances in any of these mutually independent sets is H (in the worst case, however, H could be N_i), instead of having an exponential search space of $O(exp(N_i))$, the search space grows exponentially with H and linearly with N_i (i.e., $O(N_i exp(H))$), since the solution space for each independent set can be searched independently. This reduction of search space from exponential to linear growth makes the online searching of the solution space practical. For example, for the set $\{F_R(box, robot), F_R(goal, robot)\}$, as there are

no uninstantiated referents, the independence relationship trivially holds. Hence, the set can be divided into two independent sets and each set contains one of the information instances. To obtain the potential solution to retrieve the two information instances, we can independently search through the solution space for each and simply combine the chosen solutions for both as the final potential solution. On the other hand, $\{F_R(box, X), F_R(goal, X)\}$ cannot be divided and every possible combination of potential solutions (hence exponential) from the solution spaces for retrieving each information instance must be searched to cover the original solution space, as one can influence the other through the referent of X (i.e., X must be instantiated to the same entity).

B. Semantic Reasoning

The previous section discussed the independence relationships between the information instances to reduce the search space. For further improvements, we begin by studying the semantics expressed using information instances in this section. This serves as the basis for our discussions in the following sections. First of all, we notice that in IQ-ASyMTRe, information instances of different types express disjoint semantic meanings, as the semantic meanings of different information types are naturally separated.

Lemma 3.1: Given that information instances of different types have disjoint semantic meanings, the semantic meaning related to any information requirement (i.e., specifying the required input information instances for a particular MS) can be expressed using only the logic operators (AND, OR).

Proof: Given the fact the information requirement conveys admittance rather than denial, the *NOT* operator would naturally be absent. As a result, we can represent the semantic meaning for any information requirement with the following process using only (*AND*, *OR*). The process starts by listing the maximum required number of information instances of the same type for all relevant information types (with all referent uninstantiated), connected using *AND*. Then for each possible way to fully instantiate these information instances (using different entities), we check to see if the set of information instances satisfies the requirement. This makes sure that all possible input information that may potentially satisfy the requirement would be checked. Finally, we simply need to use *OR* to connect all sets of information instances that satisfy the requirement. We call this constructed form as the *perfect-instantiated* form. Clearly, it is also in a disjunctive normal form. ■

However, expressing any semantic meaning using the perfect-instantiated form may lead to very large representations, due to the size of the entity set. (For generality, sometimes we may want to assume that the entity set is countably infinite.) The fact that IQ-ASyMTRe allows the referents of information instances to be uninstantiated provides a more concise way to express the semantic meanings. For example, $F_G(X)$ represents the semantic meaning of the global position information of any entity. In cases we need to exclude specific entities from the uninstantiated referents, they can be considered as exceptions in a final validity check.

Lemma 3.2: The semantic meaning related to any information requirement can be expressed by IQ-ASyMTRe exactly. Furthermore, assuming that the entity set is countably infinite and the information requirement has a finite representation, the finite set of information instances required for the exact expression is always the same.

Proof: The proof is a constructive one. Once we obtain the perfect-instantiated form (which is unique by construction) for the information requirement, we simply combine the conjunctive clauses (including single information instances) that can be combined using uninstantiated referents (assuming that we can check whether the entire entity set is covered or not). The combined conjunctive clauses join the combination processes immediately. When there are no more distinct combinations possible, the process terminates and we simply remove any conjunctive clauses that have been combined at least once in the combination processes. Note that the same conjunctive clauses can be used multiple times. Finally, we can simplify each conjunctive clause to remove information instances that have no instantiated referents and no referent instantiation constraints (i.e., such information instances would not be informative). As the process performs all possible distinct combinations, the result must be the most concise representation and hence the representation must be finite. Note that at any state prior to the termination, the representation would be infinite if the entity set is countably infinite. Furthermore, as the process only terminates when there are no more distinct combinations, and as the perfect-instantiated form is unique, the final set of information instances must also be unique. Hence, the conclusion holds. ■

One important note is that the representation of any information requirement in IQ-ASyMTRe, created from the process above, is still in a disjunctive normal form. Hence, each conjunctive clause independently represents a sufficient condition for a solution.

C. Removing Unnecessary Solutions

In this section, we introduce *information instance set* and *information source set*, which can represent the conjunctive clauses in the previous section and are used in our discussion on how online efficiency can be further improved.

Definition 3.2: Information Instance Set – An information instance set (IIS) is a set of information instances, and can be represented as $\{F_1, F_2, F_3, ...\}$, in which each F_i represents an information instance (for conciseness, associated referents are not shown). All information instances within the set are assumed to be connected using the AND operator.

Definition 3.3: Information Source Set – An information source set (ISS) is an IIS which also shows the sources of the information instances. An ISS can be represented as $\{ES_1{:}F_1, ES_2{:}F_2, ..., CS{:}F_{M+1}, CS{:}F_{M+2}, ...\}$. $F_{[1:M]}$ represents information instances that must be retrieved using local sensors and $F_{[M+1:...]}$ represents information instances that must be communicated.

In the following discussions, we require the IISs and ISSs to only include distinct information instances. Note

that the independencies of information instances serve to reduce the search space without changing the number of solutions encoded in the solution space. To further improve the online efficiency, we notice that another issue with the IQ-ASyMTRe solution space is the inclusion of unnecessary potential solutions. To remove them, it is not difficult to see that the leaf nodes (i.e., nodes at the farthest end from the MS) of any potential solution necessarily comprise an ISS, as the ultimate information source can only be an ES or CS. Furthermore, each ISS fully describes the interactions with other entities through specifying the information instances to be communicated. Hence, potential solutions with the exact same ISS are equivalent for task solving, although they may have different local schema connections. Hence, each distinct ISS describes a set of potential solutions and all but one can be removed from the solution space without losing solutions.

Unnecessary potential solutions can also be introduced by specific information conversions (e.g., the 2nd RPS in Table I) which only switch the ordering of the referents; on the other hand, these information conversions are necessary for more flexibility. Another note is that referent instantiation constraints are also important to determine whether a solution is necessary or not. For example, one of the potential solutions of $\{CS{:}F_G(X), CS{:}F_R(X, local)\}$ and $\{CS{:}F_G(Y), CS{:}F_R(local, Y)\}$ can be used to represent both, as X and Y have exactly the same referent instantiation constraint in both ISSs. Note that $\{CS{:}F_G(X), CS{:}F_R(X, local), CS{:}F_R(X, box)\}$ and $\{CS{:}F_G(Y), CS{:}F_R(Y, local), CS{:}F_R(X, box)\}$, on the other hand, represent different potential solutions, since they have different referent instantiation constraints.

Lemma 3.3: For any two potential solutions, if they have the same number of distinct labels (including instantiated referents) and for all distinct labels in one, we can sequentially find a matching label not previously matched in the other with the same set of information types having the same instantiated referents and the same referent instantiation constraints, then only one of them is necessary.

Proof: The proof is quite obvious since we can simply replace the labels of the uninstantiated referents in one with those of the matching uninstantiated referents in the other, respectively. The equivalence implies that if there is a solution for one, it would also be a solution for the other. ■

The independence of information instances can also be utilized to reduce the computational cost for removing unnecessary potential solutions. Suppose that the IIS related to the information requirement for a desired motor schema has N_i information instances and can be divided into K mutually independent sets, with $N_i^{[1:K]}$ information instances, respectively. The maximum number of information instances in any of these sets then is $H = Max(N_i^{[1:K]})$. Suppose that the numbers of potential solutions (i.e., size of the solution space) for retrieving these sets are $R_{[1:K]}$ respectively. The computational cost for removing unnecessary solutions would be $O(\Sigma_j {R_j}^2) = O(N_i exp(H)^2)$ (i.e., every unchecked potential solution is to be compared with the ones that have already been checked for necessity, hence the squared term) according to our discussion in the previous section, while the computational cost without taking advantage of the independencies would be $O(\Pi_j {R_j}^2) = O(exp(H)^{2N_i})$. In the experimental section, we show the practical computational savings achieved using this approach.

D. Towards More Complete Solution Spaces

As we have seen previously, for IQ-ASyMTRe to accomplish a task, the desired MSs require certain input information instances to be satisfied in order to be activated. The input information associated with the MSs can significantly influence the task execution. For one particular MS, there may be several possible options for the input information; how to choose from them so that the created solution space is more complete remains an issue. For example, one possible set of input information instances for the *go-to-goal* MS is $\{F_G(robot), F_G(goal)\}$. Another possible set of input information instances, if we have an overhead camera system, is $\{F_R(X, robot), F_R(X, goal)\}$ (i.e., we can use any camera X). As another example, for the *push-box* MS, one possible set of input information instances is $\{F_G(robot), F_G(goal), F_G(box)\}$ and another is $\{F_R(box, robot), F_R(goal, robot)\}$. How to choose the proper set of input information instances for the MSs in different situations is equivalent to the problem of generating a sufficiently complete solution space that contains the alternative solutions.

In this section, we study the relationships between IISs. A relational operator ($\succ$) is defined and used to describe relationships between them. By studying the relationships, we are able to answer how options of the input information should be chosen to provide the maximum flexibility. Note, however, that these options can be ambiguous to determine the information requirement. For example, $\{F_G(r_A)\}$ and $\{F_G(r_B)\}$ can be options for $\{\{F_G(r_A)\}$ *OR* $\{F_G(r_B)\}\}$ or $\{\{F_G(X)\}\}$. To avoid this ambiguity, we require the options of the input information to be unambiguously specified such that if an uninstantiated referent for an information instance suffices, no options should restrict the referent to be instantiated. Otherwise, even if we know from our experience that $\{F_G(r_i), i \in [1 : Z]\}$ (for a large Z) are all sufficient, we still cannot claim that $\{F_G(X)\}$ is the exact requirement.

Definition 3.4: Reduction of IIS – For any two IISs, s_1 is said to be reducible to s_2 (denoted by $s_1 \succ s_2$) if the following condition is satisfied: any information instance in s_2 is present or can be converted (i.e., using information conversions) using information instances in s_1.

Definition 3.5: Equivalence of IIS – Two IISs (s_1 and s_2) are equivalent if they satisfy the following conditions: $s_1 \succ s_2$ and $s_2 \succ s_1$.

For example, $\{F_G(r_A), F_R(r_B, r_A)\}$ is equivalent to $\{F_G(r_A), F_G(r_B)\}$ given this definition. Note that for all IISs that can be possible sets of input information instances to the desired MS, the reduction relation defines a partial ordering for the IISs. As in the examples given at the very beginning of this section, for the *push-box* MS, we have $\{F_G(robot), F_G(goal), F_G(box)\} \succ$

$\{F_R(box, robot), F_R(goal, robot)\}$, since we can use the RPS of $F_G(X) + F_G(Y) \Rightarrow F_R(Y, X)$ to do the reduction. For the *go-to-goal* MS, however, there is not a reduction relation between the two listed sets.

From Lemma 3.2, we know that any information requirement with finite representation can be specified exactly using multiple IISs connected with *OR*, represented as $\{\cup_k \text{IIS}_k\}$. As we have already mentioned, each IIS represents a sufficient and independent IIS for the requirement. We notice that the information requirement of the MS can be related to multiple unrelated methods which determine what information is required for the MS. For example, to accomplish a site clearing task, a robot can either clear the obstacles by itself or ask another teammate to come to the site to help. In the first situation, the robot needs to obtain site location and the relative positions of the obstacles while in the second, it only needs to know the site location and transfer it to the helper robot. Each method requires different information and represents different behaviors. The reduction relationship divides the IISs in the exact expression into groups, and every IIS is equivalent to the others within the same group. These different groups then naturally define the different methods as they represent different information requirements. In reality, different methods can be separately regarded by considering them as different MSs (i.e., *clear-self* and *clear-help*), so that each MS would correspond to only one particular method. Next, we further introduce the following definitions:

Definition 3.6: Power Set of IIS – The power set of any IIS s, (denoted by $P(s)$), includes all information instances that are present in s and all information instances that can be converted from s using the information conversions. An IIS that cannot produce new information instances using information conversions is called a maximum IIS (MaxIIS).

For example, for the *go-to-goal* MS, we have $P(\{F_G(robot), F_G(goal)\}) = \{F_G(robot), F_G(goal), F_R(robot, goal), F_R(goal, robot)\}$, and $P(\{F_R(X, robot), F_R(X, goal)\}) = \{F_R(X, robot), F_R(robot, X), F_R(X, goal), F_R(goal, X), F_R(robot, goal), F_R(goal, robot)\}$. Notice that the power set of any IIS is also a MaxIIS according to this definition.

Definition 3.7: Kernel IIS – For any IIS s, the kernel IIS (denoted by $K(s)$), can be any subset of information instances in $P(s)$ which satisfies: any information instance in $K(s)$ cannot be converted from any other information instances in the $K(s)$ and $K(s) \succ s$.

For example, for the *go-to-goal* MS, we list one possible kernel IIS: $K(\{F_G(robot), F_G(goal), F_R(goal, robot), F_R(robot, goal)\}) = \{F_G(robot), F_G(goal)\}$.

Definition 3.8: Minimum IIS – For all IISs that can satisfy the information requirement of a MS, the minimum IIS (MinIIS) (denoted as s_{min}) is a MaxIIS which satisfies: for any IIS s that satisfies the information requirement of the MS, we have $s_{min} \subseteq P(s)$ and s_{min} satisfies the requirement.

Now the question is whether the MinIIS for arbitrary MSs exists or not. For the purpose of practicality, we restrict our attention to MinIISs with finite representations.

Theorem 3.4: For any MS, given that the exact information requirement has a finite representation, the MinIIS with a finite representation exists and is unique.

Proof: Let us first prove its uniqueness. Suppose that there are two MinIISs for a MS, denoted by s_1 and s_2. From the definition, we can easily conclude that $s_1 \subseteq P(s_2)$ and $s_2 \subseteq P(s_1)$. As both are also MaxIISs, we have $s_1 \equiv s_2$.

From Lemma 3.2, we know that a unique set of information instances can be used to express the information requirement of a MS exactly. It follows that the IIS created (denoted by s^*) by connecting this unique set using *AND* is the MinIIS for the MS. First of all, s^* clearly satisfies the information requirement and s^* must be a MaxIIS or otherwise it would not be unique. Furthermore, according to our previous dicussions, we can conclude that all IISs for expressing the information requirement are equivalent in the exact expression. Hence, they must also be equivalent to s^*. Moreover, since the exact expression encodes all possible solutions, any IIS s satisfying the information requirement must be able to reduce to one of the IISs in the exact expression. Hence we have $s^* \subseteq P(s)$ and s^* is the MinIIS. ■

Corollary 3.5: Expressing the information requirement using any kernel set of the MinIIS maximizes the number of distinct potential solutions.

Proof: First, it is easy to see that any kernel IISs of the MinIIS are equivalent. Hence, any input information that satisfies one would also satisfy the others. Furthermore, for any IIS s satisfying the information requirement, according to Theorem 3.4, we have $\tilde{s} \subseteq P(s)$ ($\tilde{s}$ is any kernel IIS of the MinIIS). Hence, any input information that satisfies s also would satisfy $\tilde{s}$. Hence, the number of potential solutions with $\tilde{s}$ cannot be less than any IIS for the MS. ■

Now we know that expressing the information requirement using any kernel set of the MinIIS is the best choice, but how do we find the MinIIS? For simple cases, the MinIIS is obvious to determine. However, for more complex cases, it is not so obvious. Although it is not difficult to prove that finding the MinIIS using known options of the input information that satisfy the information requirement is impossible (i.e., when the same information instance is arbitrarily added to all known options but not used, there is no way to identify that the unused information instance should not be in the MinIIS). However, if we know multiple options of the input information that satisfy a MS, we can approximate the MinIIS, using Algorithm 1.

Algorithm 1 Approx. the $MinIIS$ using known options

for all $IIS_i \in$ known options **do**
  Compute $S_i = P(IIS_i)$.
end for
return $S = \cap_i(S_i)$.

For the *go-to-goal* MS, the algorithm would output $\{F_R(goal, robot), F_R(robot, goal)\}$. Interestingly, without any knowledge of the MinIIS for the MS, the algorithm actually outputs the MinIIS (though the

TABLE II

SOLUTIONS WITH THE FIRST THREE RPS'S IN TABLE I

1. Laser:$F_G(local)$
2. Fiducial:$F_R(X, local)$, CS:$F_G(X)$
3. CS:$F_G(X)$, CS:$F_R(local, X)$
4. CS:$F_G(X)$, CS:$F_R(X, local)$
5. CS:$F_G(X)$, CS:$F_R(local, X)$
6. CS:$F_G(X)$, CS:$F_R(local, Y)$, CS:$F_R(X, Y)$
7. CS:$F_G(X)$, CS:$F_R(X, Y)$, CS:$F_R(local, Y)$

system cannot determine whether it is the MinIIS) in this case. For the *push-box* MS, after applying the algorithm, we can obtain the approximated MinIIS, $\{F_R(r_A, box), F_R(box, r_A), F_R(r_A, goal), F_R(goal, r_A), F_R(box, goal), F_R(goal, box)\}$. In this case, it is also the MinIIS for the MS. Actually for the *push-box* MS, one of the inputs itself is a kernel IIS of the MinIIS.

IV. SIMULATION & EXPERIMENTAL RESULTS

We demonstrate the importance of our approach for creating complete solution spaces that are efficiently searchable by applying it to several applications in simulation and with physical robots.

A. Simulations

First, we show how the removal of unnecessary potential solutions can improve the online performance. For a simple demonstration, we first compare the potential solutions for the *go-to-goal* MS in the robot navigation task as presented in [9] using only the first three RPSs in Table I. In this task, the robot is assumed to know the global goal position and have a laser sensor for localization and a fiducial sensor for sensing its relative positions to other entities. The result is shown in Table II, in which the potential solutions shown in red are unnecessary potential solutions that are removed based on our approach. For this simple case, to find solution 6 after the removal, the reasoning process needs to process only 4 solutions instead of 6 each time it is triggered online.

Let us now look at more complicated situations. We first study the influence of the number of RPSs on the solution space. For this reason, we add in another new RPS (i.e., which increases N_c) as shown in Table I (the 4th RPS). As more information conversions allow more flexibility, the number of potential solutions would also increase. Another influential factor is the number of required information instances for the task. Table III provides the number of potential solutions (PoSs) before and after removal for various configurations for MSs in different tasks.

In Table IV, we compare the solution spaces for choosing different sets of inputs for different MSs. All information conversions in Table I are used. We can clearly see the effects of choosing the kernel IIS of the MinIIS on the solution spaces: more potential solutions are encoded such that the solution spaces for satisfying the MSs are more complete. Furthermore, it is clear that the number of potential solutions grows exponentially with the number of

TABLE III

INFLUENTIAL FACTORS FOR THE SOLUTION SPACE

Go-to-goal : RPSs Used	# PoSs	After rem.
$F_G(local)$: 1 - 3	7	4
$F_G(local)$: 1 - 4	9	5
$F_G(local)$, $F_G(goal)$: 1 - 3	14	8
$F_G(local)$, $F_G(goal)$: 1 - 4	18	10
Push-box : RPSs Used		
$F_G(local), F_G(box)$: 1 - 3	14	8
$F_G(local), F_G(box)$: 1 - 4	18	10
$F_G(local), F_G(box), F_G(goal)$: 1 - 3	28	16
$F_G(local), F_G(box), F_G(goal)$: 1 - 4	36	20

TABLE IV

MINIIS AND INDEPENDENCE OF INFORMATION INSTANCE

Go-to-goal	# PoSs	After rem.	Use Ind.
$F_G(local)$, $F_G(goal)$	18	10	7
$F_R(goal, local)$	31	15	15
Push-box			
$F_G(local), F_G(box), F_G(goal)$	36	20	9
$F_R(box, local), F_R(goal, local)$	961	185	30
Time for a full search (s)	15.1	2.9	0.02

the required input information instances to be reasoned. Without taking advantage of the independence of information instances, even after the removal of unnecessary solutions, a significant number of potential solutions would still have to be searched. If we take advantage of the information independence, the number of potential solutions required to be searched can be greatly reduced. This effect can be seen from the last row of Table IV, which also shows the time for a full search of the potential solutions for the *push-box* MS. Information independence can also be used to reduce the computational cost for removing unnecessary solutions from the solution space. In our simulations, the time to reduce the number of potential solutions from 961 to 185 (Table IV) is approximately 14.8s, while the time to reduce from 961 to 30 solutions is merely 0.02s (not the search time reported in Table IV) when taking advantage of the information independence. All times are wall-clock times and the computation is based on a 2.4GHz Core 2 Duo laptop with 2GB memory. We can see that the coupling of the two approaches can work together to achieve a much more manageable search space for certain tasks.

B. Physical Robot Experiments

For the physical robot experiments, we create several scenarios for the cooperative robot box pushing task to show how the MinIIS can benefit task execution in different situations. For this set of experiments, we choose two sets of inputs of the *go-to-goal* MS for comparison. The first set is chosen arbitrarily as $\{F_G(local), F_R(box, local), F_G(goal)\}$ (we replace $F_G(box)$ with $F_R(box, local)$ because the latter is easier to obtain using laser scan matching); the second set is an kernel

TABLE V
ARBITRARY DECISION VS. MINIIS

Configurations	Arbitrary Decision	Using MinIIS
1. Localize, Global Goal	All retrievable	All retrievable
2. Localize	All retrievable	All retrievable
3. No Localize	No $F_G(goal\|local)$	All retrievable
4. Blocked, Int. Localize	All retrievable	All retrievable
5. Blocked, Int. No Localize	No $F_G(goal\|local)$	All retrievable

IIS of the MinIIS, $\{F_R(box, local), F_R(goal, local)\}$, which is chosen according to the MinIIS approach.

We start with the simplest scenario (shown in Figure 2(a)), in which the pusher robots know where the global goal position is and can localize themselves. For the arbitrary decision approach, the global positions are directly known or can be sensed, while for the MinIIS approach, the $F_R(goal, local)$ can be computed based on the global positions using the 4th RPS in Table I. For both approaches, the $F_R(box, local)$ is obtained using laser scan matching.

Now we start imposing more constraints. If the global position is not known and must be observed (Figure 2(b)), robots utilizing both approaches can still obtain the required information. The global position of the goal can be computed using its relative position to the robot for the arbitrary decision approach, while $F_R(goal, local)$ is directly retrievable using the camera for the MinIIS approach. However, what if the pusher robots cannot localize themselves? No solution for the arbitrary decision approach is feasible (i.e., neither $F_G(local)$ nor $F_G(local)$ can be retrieved) while the MinIIS approach is still successful.

In the next scenario (Figure 2(c)), we further block the view of the goal from one of the pusher robots. However, the blocked pusher robot can see an intermediate robot teammate that can detect the goal. If the robot teammate can localize itself, then both approaches can retrieve the required information instances by requesting information from the intermediate robot. However, the intermediate robot may not have a localization capability. The arbitrary decision approach again would not be able to find any feasible solutions; on the other hand, for the MinIIS approach, the blocked pusher robot reasons out that $F_R(goal, local)$ can be obtained using $F_R(goal, other)$ and $F_R(local, other)$ and applying the 3rd RPS in Table I. The results for these scenarios are reported in Table V. In this experiment, we can see the importance of choosing the proper inputs for the MSs in various situations, as it leads to the discovering of viable coalitions that otherwise would not be found. More complete solution spaces are extremely important for task execution in dynamic environments with many unknown factors.

V. CONCLUSIONS

In this paper, we provide some improving techniques for the IQ-ASyMTRe architecture. First of all, the complexity of the solution space for IQ-ASyMTRe is analyzed; we have shown that for tasks having certain properties, the exponential growth of the number of potential solutions that have to be searched can be eliminated, instead resulting in a linear search space. This fact makes the online generation and searching of the solution space practical. Unnecessary potential solutions are also discussed and an efficient method is proposed to remove them to further improve online efficiency. Finally, the MinIIS for a task is introduced and is shown to provide a complete solution space for arbitrary tasks. We present results that illustrate the importance of these approaches for achieving online reasoning that can find dynamic solutions for tightly-coupled multirobot tasks. Although these improvements are discussed with respect to IQ-ASyMTRe, they are also applicable to architectures that approach the generality that IQ-ASyMTRe achieves.

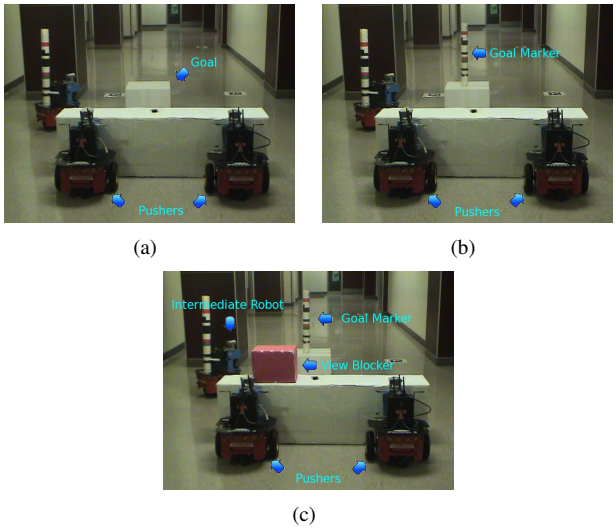


Fig. 2. (a) A box pushing scenario where the global position of the goal is known. (b) A box pushing scenario in which the goal must be observed. (c) A box pushing scenario in which the visibility of the goal is blocked. In all scenarios, the barcode markers can be detected to extract the relative position information using cameras.

REFERENCES

[1] R.C. Arkin. Motor Schema – Based Mobile Robot Navigation. *The Int'l Journal of Robotics Research*, 8(4):92–112, August 1989.

[2] B.R. Donald, J. Jennings, and D. Rus. Information invariants for distributed manipulation. *The International Journal of Robotics Research*, 16(5):673–702, 1997.

[3] C.H. Fua and S.S Ge. COBOS: Cooperative backoff adaptive scheme for multirobot task allocation. *IEEE Transactions on Robotics*, 21(6):1168–1178, 2005.

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

[5] N. Kalra, D. Ferguson, and A. Stentz. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Proc. of the IEEE Int'l. Conf. on Robotics and Automation*, 2005.

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

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

[8] Y. Zhang and L.E. Parker. A general information quality based approach for satisfying sensor constraints in multirobot tasks. In *IEEE International Conference on Robotics and Automation*, 2010.

[9] Y. Zhang and L.E. Parker. IQ-ASyMTRe: Synthesizing coalition formation and execution for tightly-coupled multirobot tasks. In *IEEE/RSJ Int'l Conference on Intelligent Robots and Systems*, 2010.