Proc. of 12th International Conference
on Advanced Robotics, Seattle, WA, 2005.

Coalescent Multi-Robot Teaming through ASyMTRe: A Formal Analysis

Fang Tang and Lynne E. Parker
Distributed Intelligence Laboratory, Department of Computer Science,
University of Tennessee, Knoxville, TN 37996-3450 Email: {ftang|parker}@cs.utk.edu

***Abstract*— This paper describes a general approach for automatically synthesizing task solutions for heterogeneous robot teams. In particular, our approach enables multiple robots to coalesce into teams to solve a task through tightly-coupled sensor sharing. Instead of designing special solution strategies for the team, our ASyMTRe approach enables the robot team to generate solutions autonomously according to the current robot team composition. In this paper, we first formulate the problems that the ASyMTRe approach addresses, and then present the anytime ASyMTRe configuration algorithm. We prove that the configuration algorithm is correct, and is guaranteed to find the optimal solution given enough time. Empirical results are also presented validating this analysis, and showing that the ASyMTRe configuration algorithm has good scalability and can quickly find a good solution with the solution quality increasing as additional planning time is available. By analyzing the configuration algorithm, we show that ASyMTRe is applicable to a large class of challenging multi-robot problems.**

I. Introduction

A challenging class of multi-robot applications requires multiple robots to coalesce into teams to solve a single task by coordinating the sharing of sensory and effector capabilities. In the most difficult of these problems, the method for solving the task, sometimes called *task decomposition*, is dependent on the available capabilities of the multi-robot team, and thus cannot be specified *a priori*. In these cases, the team must determine how to solve the task by coupling the appropriate sensory and effector capabilities from each robot, perhaps resulting in actions for each robot team member that are tightly coupled with those of other team members. Even more challenging, one or more robot team members could act based upon sensory data shared from other robots, perhaps translated into its own frame of reference. In the formal multi-robot task allocation (MRTA) framework of [7], this class of problems involves "Single-Task" robots (ST), "Multi-Robot" tasks (MR), and "Instantaneous Assignment" (IA), abbreviated ST-MR-IA[1].

For such problems, it is challenging to create robot behavior code that is flexible enough to solve tasks in a variety of ways. For example, as outlined in [12], the types of behaviors needed to deploy a mobile sensor network are shown to be highly dependent upon the total robot team composition. For homogeneous robots with the ability to detect obstacles and other team members, a potential-field-based dispersion algorithm is appropriate to achieve mobile sensor network deployment (e.g., [8]). For teams involving a highly capable "mother ship" robot and many simple sensor nodes, deployment using a marsupial delivery method is appropriate. For teams with a few robots that can perform laser localization, obstacle avoidance, and visual detection of teammates, plus many simpler robots that do not have these capabilities, an assistive navigation approach is appropriate, in which the more capable robots assist in the navigation of the simpler robots (e.g., [14]).

As another motivating example, consider a simple multi-robot transportation application where a team of robots needs to navigate from a set of starting positions to a set of goal positions. Various approaches can be used to perform the task. If every robot knows its current position relative to its goal position, a straightforward approach would be to have each robot navigate independently, for example, using laser range scanner-based localization. However, if some robots do not have the sensing capability to localize, an alternative approach would be for the more capable robots to guide the less capable robots towards their goals by providing them with positioning information, for example, using a camera to estimate the relative position of another robot. Alternative approaches could be imagined in other team compositions. The important point here is that the resulting robot behaviors for accomplishing the task could be different depending upon the combination of sensors that is available for solving the task. Ideally, we would like to design the behavior code for an individual robot to be independent of any particular robot team composition, towards the goal of flexible, reusable robot behaviors.

To address this class of multi-robot problems, we are developing an approach, called ASyMTRe (Automated Synthesis of Multi-robot Task solutions through software Reconfiguration, pronounced "Asymmetry"), which provides general mechanisms for multi-robot teams to: (1) synthesize single task solutions based upon the current team composition, (2) share sensory information across networked robots, as required to accomplish the task, and (3) automatically instantiate these task solutions directly into executable robot code [17]. This approach is a type of *coalition formation* (e.g., [15]), which has been extensively studied in the multi-agent community, and is also beginning to be addressed in the multi-robot community (e.g., [18]). However, our work goes beyond the mapping of agent/robot capabilities to tasks and finding the best combination of agents/robots for the task. ASyMTRe

[1]However, note that unlike the multi-robot taxonomies of [7] and [2], we believe that it is misleading to independently label a task as either "single-robot" or "multi-robot", without taking into account the robot team capabilities. In reality, the number of robots required to perform a task is dependent on the mix of capabilities of the available robots, even in the same application.

defines mechanisms based on schema theory ([10] and [1]) and the mapping of schemas to information types (inspired by [5]) to enable robot team members to share information across robots, and to enable robots to automatically instantiate the derived team solution into executable code on the robot team members.

In our prior work on ASyMTRe ([17], [13], [4]), we demonstrated a proof-of-principle application on physical robots that illustrates the ability of ASyMTRe to generate executable code enabling two robots to share sensory data for performing a *go-to-goal* task. Additionally, this prior work illustrated the ability of ASyMTRe to generate multiple solutions to the *box-pushing* problem using different robot team compositions.

This paper builds on this prior work to provide a theoretical analysis of the ASyMTRe approach in terms of soundness, completeness, and optimality, and presents empirical results showing the scalability of this approach. These results allow us to provide evidence of the applicability of this approach to a large class of challenging multi-robot problems.

The remainder of this paper is organized as follows. Section II describes the ASyMTRe solution approach. Section III analyzes the theoretical soundness, completeness, and optimality of this approach. Section IV describes simulation results that empirically verify the analysis and illustrate the scalability of the approach. We briefly outline related work in Section V, and provide concluding remarks in Section VI.

II. The ASyMTRe Approach

In this section, we briefly outline the ASyMTRe approach to provide the foundation for our theoretical analysis (see [17] for a more detailed presentation of the approach).

A. Schema Theory and Information Types

Our approach is based on a distributed extension to schema theory ([1] and [10]). As such, the basic building blocks of our approach are collections of environmental sensors (ES), perceptual schemas (PS), motor schemas (MS), and a simple new component we introduce, called *communication schemas* (CS). Perceptual schemas process input from environmental sensors to provide information to motor schemas, which then generate an output control vector corresponding to the way the robot should move in response to the perceived stimuli. Communication schemas transfer information between schemas distributed across multiple robots, which are introduced to distinguish the connections built within a robot from the connections across robots. All of these schemas are assumed to be pre-programmed into the robots at design time, and represent fundamental individual capabilities of the robots.

ASyMTRe allows robots to reason about how to solve a task based upon the fundamental information needed to accomplish the task. The information needed to activate a certain schema remains the same regardless of the way that the robot may obtain or generate it. Thus, we can label the input and output of all schemas with a set of information types that are unique to the task. Note that we use the term *information types* as distinct from *data types*. Semantics of the information is built into these information types, and does not just refer to a data type (such as boolean or integer). For example, the input information types of a *go_to_goal* schema could be {*current_position, goal_position*}, and its output types could be the specific motor commands. We define the inputs and outputs of the schemas to be the set of information types $F = \{F_1, F_2, ...\}$. For schema S_i, I^{S_i} and $O^{S_i} \subset F$, represent the set of input and output of S_i, respectively. Like in [10], we assume that each schema has multiple inputs and outputs. As shown in Fig. 1, there are two types of inputs to a schema. The solid-line arrows going into a schema represent an "OR" condition, meaning that it is sufficient for the schema to only have one of the specified inputs. The dashed-line arrows represent an "AND" condition, where all the indicated inputs are needed to produce an result. For example, MS_1^1 can calculate output only if it receives both F_1 and F_2. However, PS_2^1 can produce output based on either the output of PS_1^1 or CS_1^1. An output of a schema can be connected to an input of another schema if and only if their information labels match. Using the mapping from schemas to information types, robots can collaborate to define different task strategies in terms of the required flow of information in the system. Once the interconnections between schema are established, the robot team members have executable code to accomplish their task.

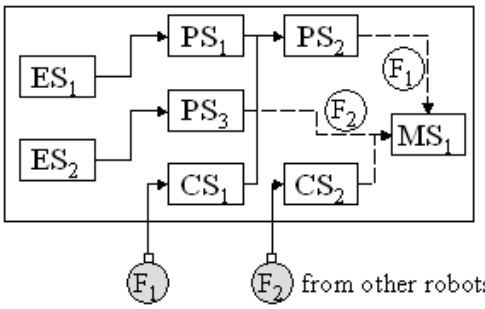

Fig. 1. An example of how the schemas are connected to accomplish a task. (According to the notation of Section II-B, here, $R_1 = (ES_1^1, ES_2^1, PS_1^1, PS_2^1, PS_3^1, CS_1^1, CS_2^1, MS_1^1)$, $T = \{MS_1^1\}$, $F = \{F_1, F_2, \cdots\}$, $I^{MS_1^1} = \{F_1, F_2\}$, $O^{PS_2^1} = \{F_1\}$, $O^{PS_3^1} = \{F_2\}$, and $I^{CS_i^1} = O^{CS_i^1} = \{F_i\}$.)

TABLE I
CONNECTION CONSTRAINTS FOR SCHEMAS

Sensor/Schema	Input Sources	Output Feeds into:
ES	Sensor Signals	PS
PS	ES, PS or CS	PS, CS or MS
CS	PS, or CS	PS, CS, or MS
MS	PS, CS, or ES	Actuators

B. Formulation of Interconnected Schemas

Given a set of n robots and a task *T*, the problem can be represented as (R, T, U), where $R = \{R_1, R_2, \cdots, R_n\}$ is the set of n robots, and U provides utility information to be defined later. T is the set of motor schemas (specific to robots) that define the team-level task to be achieved, along with application-specific parameters as needed. For example,

suppose there are n robots that are pre-programmed with an MS_1, and it is required that the team accomplish the task as a whole, we could define $T = \{MS_1^1, MS_1^2, \cdots, MS_1^n\}$. A robot, R_i, is represented by $R_i = (ES^i, S^i)$. ES^i is a set of environmental sensors that are installed on R_i, where $O^{ES_j^i} \subset F$ is the output of ES_j^i. S^i is the set of schemas that are pre-programmed into R_i at design time. Each schema is represented by $(S_j^i, I^{S_j^i}, O^{S_j^i})$. A schema can be activated if and only if its input can be obtained from the output of another schema or sensor. Specifically, a set of *Connection Constraints* are used to regulate the connections between schemas. As shown in Table I, the constraints specify the restrictions on correct connections between various schemas.

Given the above information, the problem (R, T, U) has a solution if and only if for each $R_i \in R$ and for all $MS_j \in T$, the inputs of MS_j are satisfied, along with all the inputs from the schemas that feed into MS_j. A solution is a combination of schemas that can satisfy the above requirements. Fig. 1 shows how schemas can be connected to each other to compose a possible solution.

With multiple solutions available, we need to measure the quality of each solution. Most of the prior work assumes a generic utility/cost function without specifying the implementation details [15]. Here, we attempt to formulate the utility function by instantiating the generic definition of utility presented in [7]. We define a *sensori-computational system* (SCS) [5], which is a module that computes a function of its sensory inputs. In our system, this is represented as $SCS_j^i = (S_j^i, ES_j^i)$, where S_j^i is the jth PS/CS on R_i. Each SCS_j^i has an associated cost C_j^i and a success probability P_j^i. The cost is determined by the sensory and computational requirements of the solution. Perceptual processes with a significant amount of sensor processing, such as image processing, are given higher sensing costs. Success probability is an estimated value based upon experiences. Perceptual processes that are easily influenced by environmental factors, such as image processing under different lighting conditions, are given lower success probabilities. The utility function for R_i's contribution to the solution is defined as:

$$U_i = \sum_j (w \cdot P_j^i - (1 - w) \cdot C_j^i) \quad (1)$$

where w is the weight of the success probability relative to the sensing cost.[2] The metric sums over all the SCS_j that R_i needs to activate to accomplish a task. The utility of a complete solution is the sum of the individual robot utilities, which is $U = \sum_i U_i$. In summary, the problem we study here is: Given a problem (R, T, U), what is the solution (consisting of a distributed configuration of schemas) such that the total utility U is maximized?

[2]In fact, the utility of a solution should also consider other aspects, such as the quality of information, frequency of the output, the computational complexity, etc. We will extend our utility definition to include these aspects in future work.

C. Potential Configuration Space (PCS)

When a group of robots is brought together to accomplish a task, each with its unique sensors and corresponding schemas, one way to solve the problem is to perform an exhaustive search of all the possible combinations of schemas within or across robots. We refer to this complete space as the Original Configuration Space (OCS). However, the number of possible connections in the OCS is exponentially large in the number of robots. In the general case, the robots will be developed separately, and it is highly possible that the schemas with the same functionality are represented differently on different robots. By definition, schemas S_i and S_j are of the same functionality, *func(S_i) = func(S_j)*, and are thus in the same equivalence class, if and only if $I^{S_i} = I^{S_j}$, $O^{S_i} = O^{S_j}$, and they have the same success probability and sensing cost. To reduce the size of the search space, we generate a reduced configuration space, called the Potential Configuration Space (PCS), by including only one entry for each equivalence class of schema. The extent of the size reduction that we achieve depends upon the specific team composition, and the degree of overlap in their capabilities. Step I in Fig. 2 shows an example of reducing the configuration space. Assume, without loss of generality, that the team is composed of n robots, each of which has h schemas, and each schema requires k inputs. The OCS is of size is $O((nhk)^{nh})$. After the reduction, the PCS is of size $O((h'k)^{h'})$, where h' is the number of equivalence classes, and $h' < nh$. This analysis assumes that every output of any schema can be a potential input of any schema. In practice, the size of the OCS and PCS are even smaller because of the connection constraints shown in Table I. The conversion from the OCS to the PCS not only reduces the size of the entire search space, but also discards the duplicated solutions. As an example, consider a case of n homogeneous robots that have the same p schemas. Then, the OCS would have np schemas, but the PCS would only have p schemas, since the duplicates would be removed from the PCS. Thus, in practice, the reduction can have a significant impact on the search space size.

Theorem 1: *If a solution exists in the OCS, it must exist in the PCS.*

Proof: Assume to the contrary that one of the solutions present in the OCS does not exist in the PCS. Recall that a solution is a combination of schemas. This could only occur if at least one schema S_i in the solution does not exist in the PCS, which means there exists an S_i in the OCS, but there does not exist an S_j in the PCS such that *func(S_i) = func(S_j)*. According to the definition of the PCS, there will be one entry for each equivalence class of schema. Thus, there cannot be a solution in the OCS that does not exist in the PCS. □

D. Instantiation on Robot Teams

After the PCS is generated, the ASyMTRe configuration algorithm searches through the PCS to generate a list of *potential solutions*, which are the exhaustive combinations of schemas (within or across the robots) that could be connected for a single robot to accomplish its part of the task. For

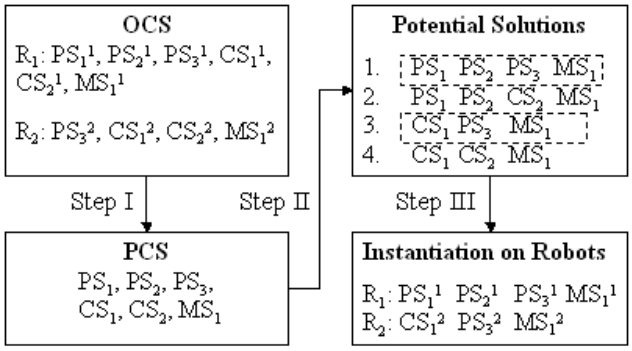

Fig. 2. Step I: Reduce the OCS to the PCS by creating one entry per schema equivalence class. Step II: Generate the list of potential solutions based on the schemas in the PCS. Step III: Instantiate the selected potential solutions on specific robots. Assume $T = \{MS_1^1, MS_1^2\}$, *func*(PS_k^i) = *func*(PS_k^j), *func*(CS_k^i) = *func*(CS_k^j), and *func*(MS_k^i) = *func*(MS_k^j).

example, suppose there are n robots that are all required to activate their motor schema MS_1. We would therefore define $T = \{MS_1\}$. Then a potential solution could be one of the combinations of schemas such that the input to MS_1 is satisfied, along with the input of other schemas that feed into MS_1. It is important to note that the list of potential solutions are not for the entire task, but are ways to connect schemas in the team of robots in order for a robot to perform the task. During the configuration process, the algorithm searches through the list of potential solutions to assign the best solution for each robot. As shown in Theorem 1, the algorithm will not miss any solutions by searching in the PCS. Once solutions are found in the PCS, we still need to map them back to the OCS and instantiate them on robots by translating the schemas in the solutions into robot-specific schemas. Steps II and III in Fig. 2 demonstrate an example of the instantiation process.

The configuration process involves greedily searching through the list of potential solutions and selecting the solution with the maximum utility from each robot's perspective. Suppose the algorithm assigns a solution to every robot R_i with an utility U_i. Our goal is to maximize the total utility $\sum_i U_i$. Similar to [15], we also assume robots work in a non-super-additive environment. Thus, the bigger a coalition is, the higher the communication and computation costs are. In practice, we impose a *maximum cooperation size* constraint on the algorithm, to reduce the complexity of the robots executing the resulting solution. When assigning potential solutions, the exhaustive, brute force approach would involve enumerating all possible orderings of robots and selecting the one that maximizes $\sum_i U_i$. However, this approach could be impractical since the complexity is $O(n!)$, where n is the number of robots. Therefore, we impose an additional heuristic on the search process, which is the ordering in which robots are considered by the algorithm. At the beginning, we heuristically define two special orderings with which we begin the search, to increase the likelihood of generating a quality result. The first ordering sorts robots according to increasing robot sensing capabilities. Less capable robots whose motor schema must be activated as part of the solution are considered first, because it is generally more difficult to find a solution when the sensing resources are limited. Our second ordering sorts robots by the number of robots that they can cooperate with, as calculated during the first search process. This number represents the number of robots that can provide the information that is needed by the current robot. Thus, robots with fewer other robots to cooperate with will have the chance to find their collaborators earlier in the search process.

An *anytime algorithm* [20] can be applied here, since it can provide a satisfactory answer within a short time and its quality of results can be improved given more time. In our domain, the algorithm first generates all the orderings of robots with which the algorithm can configure solutions. If more time is available, another ordering of robots is selected sequentially and the above procedure is repeated until the deadline is reached. The algorithm will report the solution with the highest utility it has found so far or report no solution if a solution is not found. Since we have a finite number of robots and a finite solution space, the algorithm will ultimately find the optimal solution if there exists one, given sufficient search time. All of the previously described aspects of the ASyMTRe approach are combined to yield the centralized ASyMTRe configuration algorithm shown in Table II.

III. Algorithm Analysis

We now analyze the soundness, completeness, and optimality of the ASyMTRe configuration algorithm.[3] Here, *soundness* is the proof of the correctness of the generated solutions with respect to the environmental setting. *Completeness* is the guarantee that a solution will be found if it exists. *Optimality* is ensuring that the system will select an optimal solution.

A. Soundness

Theorem 2: *The ASyMTRe configuration algorithm is correct.*
Proof: Assume to the contrary that the algorithm is not correct. This could occur in two ways: (1) At least one of the connections made is not a valid connection; (2) Not all needed connections are made to compose a solution. We argue that these two cases will not occur.

First, an output of a schema S_i can be connected to an input of a schema S_j if and only if their information types match. Since we know all the input and output information types for every schema, and we maintain the connection constraints that specify the types of schema connections that the system allows, it is not possible to generate an invalid connection.

Second, a potential solution is a combination of schemas such that the inputs for each $MS_i \in T$ are satisfied, along with all the inputs from the schemas that feed into MS_i. Since the algorithm complies with this standard, all needed connections will be made to become a potential solution. Thus a solution generated by ASyMTRe will be a correct solution. □

[3]Due to the similarity between our configuration algorithm and the coalition formation algorithm presented in [15], we also plan to analyze the bounds on our solution quality in future work. It has been proved in [15] that similar algorithms are of low logarithmic ratio bounds to the optimal solution.

TABLE II

THE ASYMTRE CONFIGURATION ALGORITHM

The ASyMTRe Configuration Algorithm ((R, T, U), Deadline)

(R, T, U): the robot team composition, task, and utility

n: the number of robots in the team
m: the number of configurations to accomplish the task
k: a constant, which specifies the number of iterations

1) Generate all sequential orderings of the robots.
2) Generate the PCS based on equivalence classes. [$O(n)$]
3) Generate a list of potential solutions (size m) by connecting schemas in the PCS to satisfy task-requirements.
4) Sort the robot team members according to increasing sensing capabilities. [$O(n\log(n))$]
5) Configure solutions on the robot team: [$O(kmn^2)$]
   - For each robot R_i in current ordering: [$O(n)$]
     - For each potential solution j: [$O(m)$]
       - If R_i can accomplish the task by itself, assign solution j to R_i. [$O(1)$]
       - Else check the other $n-1$ robots to see if one can provide the needed information. [$O(n)$]
       - If the estimated utility U_i of R_i using solution j is greater than the utility of its previous solution, and the constraints on cooperation size are satisfied, update the solution strategy on R_i. [$O(1)$]
   - Continue the above process until:
     - All robots can perform the task.
     - Or, after k number of trials.
   - Calculate the team utility *U*. If *U* is greater than the utility of previous solutions, update them.
   - If more time is given: If the special ordering (increasing number of robots that one can cooperate with) has not been checked, use this ordering; otherwise, select an ordering sequentially; repeat the above process.
6) If a solution exists, report the current best solution. Otherwise, report "Failure".

B. Completeness and Optimality

Since the heuristics cause the search to begin in a greedy manner, searching the most promising candidates first, the search process (based on the current ordering) suffers from the well-known problems of greedy algorithms. Namely, in our domain, the ordering in which the robots are considered may cause the system to miss solutions, even though sufficient sensor resources exist to accomplish the task. Such an example is shown in Fig. 3. In this example, we assume that R_3 can provide the information needed by both R_1 and R_2, and R_4 can provide the information needed by R_2. The utility for each connection is different and the maximum cooperation size is two. Consider the situation where the ordering used by the algorithm is $R_2 \prec R_1$. The algorithm certainly chooses R_3 to cooperate with R_2 because of the higher utility. Therefore, R_1 cannot find any robot to cooperate with. To avoid this situation, the algorithm reorders the robots by the potential number of robots that can help them, which is $R_1 \prec R_2$ in the example, since R_1 has one potential robot to cooperate with, while R_2 has two. With this ordering, the solution is to have R_3 cooperate with R_1 and R_4 cooperate with R_2 with a total team utility of 3.

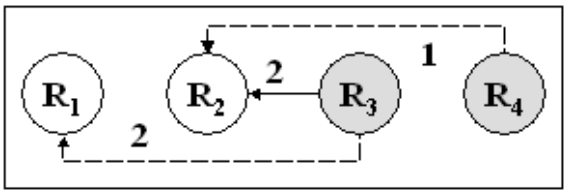

Fig. 3. A special case that might produce incomplete results. The numbers indicate utilities.

TABLE III

FACTORS THAT INFLUENCE THE ANALYTICAL RESULTS

Labels	Factors	Ranges
n	Size of the robot team	1 to 100
p	Number of inputs to a schema	1 to 10
q	Number of redundant schemas	1 to 5
m	Number of potential solutions	1 to 128

Theorem 3: *The ASyMTRe configuration algorithm is complete and optimal given enough time.*

Proof: We prove completeness and optimality by showing that the algorithm can sequentially search the entire solution space given enough time. Step 4 (in Table II) shows the major configuration process in which, given an ordering of robots, the algorithm searches through the list of potential solutions to find solutions for every team member. Note that the ordering is sequentially selected rather than randomly generated. If the algorithm is given enough time, it will ultimately test all possible orderings of robots, which is $O(n!)$, and reports the solution with the highest utility. Since the solution space is eventually explored in its entirety by the algorithm, it is complete and optimal, given enough time. □

Despite this proof, we note that in reality, practical constraints require a fast response, especially when dealing with failures that require quick reconfiguration. Since the problem itself is NP-hard and the worst case time complexity is $O(n!)$, it is not possible to generate an optimal solution in practice when n is large. However, in our experiments, the algorithm can always return a good solution for the team within a few seconds.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

In related work ([17], [13], [4]), we have demonstrated the ability of ASyMTRe to automatically generate different cooperation solutions to the same task, as a function of the robot team capabilities. Here, we illustrate the computational performance of the ASyMTRe configuration algorithm by reporting the results of a large number of "generic" experiments in simulation. The results illustrate the computational tractability and scalability of ASyMTRe in practice.

A. Experimental Setup

A variety of characteristics in an experimental setup will influence the results of ASyMTRe, such as the robot team

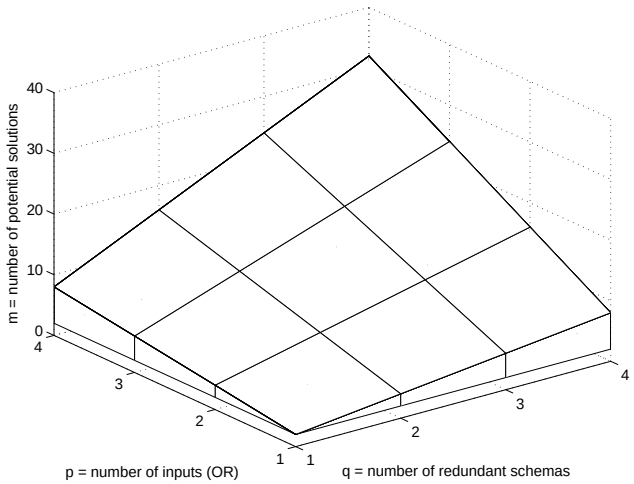

Fig. 4. The number of potential solutions is decided by the number of inputs ("OR" condition) and the number of redundant schemas. Here, n = 10.

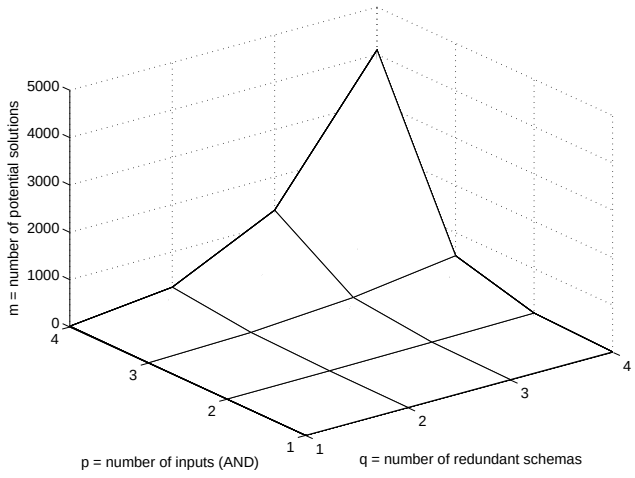

Fig. 5. The number of potential solutions is decided by the number of inputs ("AND" condition) and the number of redundant schemas. Here, n = 10.

composition and the available schemas. In Table III, we list all the major factors we considered in our applications and their ranges. The "number of redundant schemas" represents the number of schemas that can provide the same type of information within the robot team. For example, we can develop two different perceptual schemas to use either a laser scanner or a camera to estimate the relative position of an object. During the experiments, we randomly generate values for different factors and apply the ASyMTRe configuration algorithm, collecting data on the time needed to generate a solution, and the size of the list of potential solutions.

B. Potential Solutions

Since the computational complexity is partially determined by the number of potential solutions, we first explore how the characteristics of the schemas influence this size. We ran the configuration algorithm on 100 problem instances with different schemas setups by varying the number of inputs to a motor schema and the number of redundant schemas that provide the inputs to the motor schema. Fig. 4 and Fig. 5 plot the number of potential solutions as we increase the redundancy and the number of inputs for OR-type inputs and AND-type inputs respectively. Note that, the number of potential solutions is also dependent on the team size, especially when new schemas are introduced into the PCS. However, the increase introduced by the team size is relatively small compared with the number of inputs and the number of redundant schemas. Thus, we used a specific team size (n = 10) in these experiments. The curves should be similar to this case for the other team sizes.

The results are consistent with our analysis. When a schema has multiple choices of inputs (OR), the number of the potential solutions increases linearly with the increasing redundancy. If an MS needs one of the p inputs and each input can be provided by q schemas, then the number of potential solutions is $O(pq)$. When a schema needs the combination of multiple inputs (AND), the size increases exponentially with the increasing redundancy. Here, if an MS needs p inputs and each input can be provided by q schemas, then the number of potential solutions is $O(q^p)$. However, this result assumes that solution involves a large coalition of robots, all of which are sharing sensor data. In practice, this is impractical because of the constraints on the coalition size. If we impose the constraints on the coalition size, the PCS can be reduced. However, the robot's decision making process is less flexible.

These results show that there is a tradeoff between the complexity and the flexibility of a solution. Increasing the flexibility by allowing a large coalition may increase the solvability of a task but may also increase the complexity of the solution, and thus the computational requirement for finding that solution. Usually, when the flexibility is not crucial to the task, using the constraints would make the solution generation process quicker.

C. Time Complexity and Scalability

Since the ASyMTRe configuration algorithm is an anytime algorithm, we can measure the complexity of the algorithm by observing the search procedure only; that is, how long it takes to configure solutions on a team of robots given one ordering of the robots. The time complexity of the search process is $O(mn^2)$, where m is the number of potential solutions (exponential in n) and n is the number of robots. We ran this algorithm over 1,000 problem instances with different m and n values within the ranges specified in Table III. The sensors, schemas and their utility information in these experiments are arbitrarily generated to make it more general. The actual running time to find a first solution for all of the experiments ranges from 1 to 2 seconds. Experiments were run on a Linux machine with 512MB RAM and an Intel Pentium 4 2.4GHz processor.

These experiments illustrate that the running time for finding a first solution does not vary dramatically with different settings. To see how the number of potential solutions (m) and the number of robots (n) influence the complexity of the

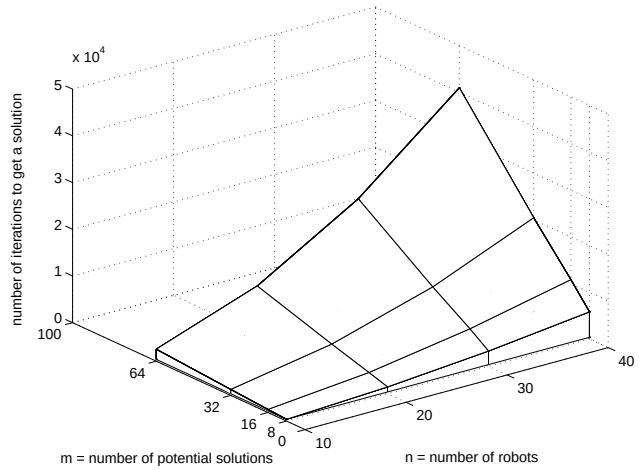

Fig. 6. Scalability of the ASyMTRe configuration algorithm

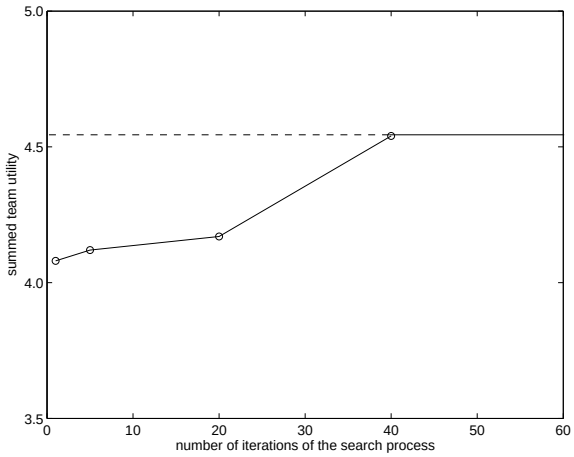

Fig. 7. The quality of the solution increases over time. The dashed-line indicates the maximum team utility.

approach, we measured the number of iterations taken for a robot team to find a solution. As suggested in [7], we also assume that the running time is determined by some dominant operation, which in our domain is the comparison of utility. Fig. 6 shows the experimental results for the scalability of the ASyMTRe configuration algorithm, in which n varies from 10 to 40, and m varies from 8 to 64. These empirical results confirm that the running time to complete a single search process is $O(mn^2)$.

To demonstrate the anytime aspect of the ASyMTRe configuration algorithm, we collected data on how the quality of the solution can be improved over time. Fig. 7 plots the relationship between time and solution quality for one of our typical runs. Here, each search process is associated with an ordering of the robots, with the special ordering (according to increasing sensing capabilities) as the starting point. The quality of the solution, which is determined by the summed team utility, increases slowly as we increase the number of iterations of the search process. Since this example is relatively simple (n = 8), we were able to find the optimal solution by listing all possible solutions. For this particular run, the optimal solution was obtained after 40 iterations, which is about 40 seconds. This graph illustrates that the algorithm can report a good solution at any time and the quality of the solution increases with more time.

D. Discussion and Summary of Results

We have applied the ASyMTRe approach to a physical team of two Pioneer robots on a transportation task [13]. Three different cooperative methods were generated to accommodate the change in robot team composition. The ASyMTRe approach coalesces robots into teams to solve a single task by coordinating the sharing of sensors and effector capabilities, and is suitable for applications where robots cooperate with each other to accomplish a team-level goal. The cooperation enables some robots to accomplish the task that is impossible for them to achieve by themselves. Since the algorithm runs in a reasonable time (1 to 2 seconds) to provide a satisfactory solution and the running time does not increase dramatically as the size of the application increases, it is applicable to most of today's multi-robot applications, where the team size is normally less than 100 robots. This feature is also important in applications where robot or sensor failures are common. Whenever there is a failure, the algorithm can be called again to synthesize new solutions, taking into account the current robot team's sensing capabilities. The ASyMTRe approach uses heuristic search to find good quality solutions in a short time, and the quality of the solution can be increased over time. However, it still takes significant time to find an optimal solution, especially for large problems. An evolutionary approach is worth considering here in our future work as an alternative method for generating good solutions.

V. Related Work

The formalism of the ASyMTRe problem is similar to the formal model defined in [10]. In [10], a robot schema includes: a list of input and output ports, a local variable list and a behavior. The ports of the same type can be connected together. At the lower level, each schema is instantiated with the proper variables. A network of schemas can then be built if we connect the output ports of a schema with the input ports of another schema. At the higher level, an assemblage of schemas can be established to represent the interaction between schemas. In this model, task plans are defined as the sensory schemas, motor schemas and the possible connections between schemas. Compared with this work, our approach dynamically connects the schemas at run time instead of using pre-defined connections. Furthermore, a schema is considered as a black box that takes certain input and generates output. We do not need to know the "behavior" – the process of generating output from the input. Schemas are not connected to each other at the beginning of a task but are instantiated after the solution strategies are generated. The interconnections of schemas also build a networked schemas to achieve the team-level goal.

Most research in heterogeneous robot teams focuses on the proper selection of robots and their actions that will result

in the collaborative achievement of team-level goals. There are two main trends for addressing this issue: *task allocation* and *role assignment*. From the task perspective, the task is decomposed into independent subtasks or hierarchical task trees either by a general autonomous planner, or simply by the designer. Robot team members then determine which robot should perform which subtask based upon the individual utility for performing various subtasks. Examples of this approach include the behavior-based architectures such as ALLIANCE [11] and BLE [19], and market-based approaches such as M+ [3], MURDOCH [6], and First-price auctions [21]. A formal analysis of the computation, communication requirements and solution qualities of these approaches is presented in [7]. From the role perspective, the application is subdivided into roles that define action strategies for achieving part of the application. Robot team members then determine which robot should perform which role (e.g., [16], [9]). Multi-agent systems research also provides a variety of approaches to coordinate agent behaviors. In these approaches, agents are organized into coalitions to achieve a higher-level goal. The motivation behind coalition formation and the ASyMTRe approach are similar in that the coalitions are formed with a purpose to accomplish the team-level goals [15]. However, there are fundamental differences between the abstractions used to represent agent/robot capabilities [18].

These prior approaches assume that the robots/agents know how to accomplish a certain task; ASyMTRe allows robots to determine how to perform a task based on the sensory and effector capabilities of the robot team and results in directly executable cooperative robot control code. Compared with these existing approaches, ASyMTRe focuses on the ST-MR-IA domain [7] and enables robot team members to share sensory information across networked robots to accomplish a task. The synthesis of these cooperative task solutions is based on the interconnection of environmental sensors, perceptual, communication, and motor schemas. The result is the coalescence of multiple robots to solve a team-level task.

VI. Conclusions and Future Work

This paper has analyzed the ASyMTRe approach from the theoretic and the experimental perspectives. The ASyMTRe approach synthesizes single task solutions based upon the team composition by enabling the sharing of sensory information across robots. We have shown that the centralized ASyMTRe configuration algorithm is sound, complete and optimal given enough time. Additionally, we have demonstrated that ASyMTRe can generate a good quality solution within 1 to 2 seconds, with the quality of the solution improving over time. Thus, this approach can be applied to a large class of challenging multi-robot problems.

Our ongoing work includes developing a distributed ASyMTRe configuration algorithm to see how the distributed decision making process influences the results.

References

[1] R. C. Arkin. Motor schema based navigation for a mobile robot: an approach to programming by behavior. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 264–271, 1987.

[2] T. Balch. Taxonomies of multi-robot task and reward. In T. Balch and L. E. Parker, editors, *Robot Teams: From Diversity to Polymorphism*. AK Peters, 2002.

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

[4] M. Chandra. Software reconfigurability for heterogeneous robot cooperation. Master's thesis, University of Tennessee, Department of Computer Science, May 2004.

[5] B. R. Donald, J. Jennings, and D. Rus. Towards a theory of information invariants for cooperating autonomous mobile robots. In *Proceedings of the International Symposium of Robotics Research*, Hidden Valley, PA, October 1993.

[6] B. Gerkey and M. J. Mataric. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 16(5):758–768, 2002.

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

[8] A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5: Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, pages 299–308. Springer-Verlag, 2002.

[9] J. Jennings and C. Kirkwood-Watts. Distributed mobile robotics by the method of dynamic teams. In *Proceedings of Fourth International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 1998.

[10] D. M. Lyons and M. A. Arbib. A formal model of computation for sensory-based robotics. *IEEE Transactions on Robotics and Automation*, 5(3):280–293, 1989.

[11] L. E. Parker. ALLIANCE: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[12] L. E. Parker. The effect of heterogeneity in teams of 100+ mobile robots. In A. Schultz, L. E. Parker, and F. Schneider, editors, *Multi-Robot Systems Volume II: From Swarms to Intelligent Automata*. Kluwer, 2003.

[13] L. E. Parker, M. Chandra, and F. Tang. Enabling autonomous sensor-sharing for tightly-coupled cooperative tasks. In L. E. Parker, A. Schultz, and F. Schneider, editors, *Multi-Robot Systems Volume III: From Swarms to Intelligent Automata*. Kluwer, 2005.

[14] L. E. Parker, B. Kannan, F. Tang, and M. Bailey. Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2004.

[15] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[16] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Simith. First results in the coordination of heterogeneous robots for large-scale assembly. In *Proc. of the ISER Seventh International Symposium on Experimental Robotics*, 2000.

[17] F. Tang and L. E. Parker. ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2005.

[18] L. Vig and J. A. Adams. Issues in multi-robot coalition formation. In L. E. Parker, A. Schultz, and F. Schneider, editors, *Multi-Robot Systems Volume III: From Swarms to Intelligent Automata*. Kluwer, 2005.

[19] B. B. Werger and M. J. Mataric. Broadcast of local eligibility for multi-target observation. *Distributed Autonomous Robotic Systems 4*, pages 347–356, 2000.

[20] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

[21] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3016–3023, 2002.