8-2012

# Coalition Formation and Execution in Multi-robot Tasks

Yu Zhang
yzhang51@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Yu Zhang entitled "Coalition Formation and Execution in Multi-robot Tasks." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Lynne E. Parker, Major Professor

We have read this dissertation and recommend its acceptance:

Michael W. Berry, Bruce MacLennan, Peiling Wang

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by Yu Zhang entitled "Coalition formation and execution in multi-robot tasks". I have examined the final paper copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Lynne E. Parker, Major Professor

We have read this dissertation
and recommend its acceptance:

Michael W. Berry

Bruce MacLennan

Peiling Wang

Accepted for the Council:

Vice Chancellor and Dean of
Graduate Studies

# Coalition Formation and Execution in Multi-robot Tasks

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Yu Zhang

August 2012

# Dedication

This dissertation is dedicated to my parents, Wenxiang Zhang and Lanqiong Yu, who have always been there to support me. They encouraged me when I felt most depressed and embraced my happiness when I succeeded. Without their support and understanding, I could not have made solid progresses towards achieving my life goals. I also want to dedicate this dissertation to my grandparents, Gaoyou Zhang, Shiguang Qian and Zuorun Yu, Shuzhen Qian, and other members in my family. Finally, I want to dedicate this dissertation to my girl friend, Kai Sha, who provided tremendous encouragement and support to me to complete this work.

# Acknowledgments

This dissertation could not have been completed without inspiration and help from many people. First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Lynne E. Parker. It was her great inspiration and guidance that the work in this dissertation was based on. Her professionalism and willingness to guide students to become good researchers led to important progresses for me towards achieving my academic goals, which are indispensable to the completion of this dissertation. I would also like to thank the rest of my committee, Dr. Michael W. Berry, Dr. Bruce MacLennan and Dr. Peiling Wang, for their willingness and dedication to serve on my committee, and for their advices and support in improving and accomplishing this work.

I additionally would like to thank other faculties and staff in the Center for Intelligent Systems and Machine Learning (CISML) and in the Electrical Engineering and Computer Science (EECS) department for their great support. In particularly, I would like to thank Dr. Hairong Qi, Dr. Itamar Arel, Dr. Husheng Li, Mr. Scott Wells, Ms. Dana Bryson, Ms. Julia Elkins and Ms. Tiffany Harmon.

I would also like to thank my fellow graduate students, Xiangyan Li, Yuanyuan Li, Dwi Sianto Mansjur, Richard Edwards, John Hoare, Hao Zhang, Sudarshan Srinivasan, Mike Franklin, Chris Reardon, Robert Lowe and Nick Overfield, for their heartily support during my study towards this dissertation. I also enjoyed working with the robots in the Distributed Intelligence Laboratory (DILab), including Arno, Blitz, Cappy and Duke, although it often required serious efforts to have them cooperate.

Finally, I must express my appreciation to many other friends who have accompanied me during these years. They have made the intense life of graduate studies much more enjoyable and relaxing, which is definitely a determinant factor to accomplish anything that demands such extreme energy and efforts. I would like to thank Tabitha Samuel, Liangcheng Yang, Yulu Jia, Teng Ma, Xiaolan Chen, Xia Huang, Hao Tang, Zhiqiang Li, Hui Lin, Zhi Han, Cherry Xu, and many others who have always been there for me whenever needed most.

# Abstract

In this research, I explore several related problems in distributed robot systems that must be addressed in order to achieve multi-robot tasks, in which individual robots may not possess all the required capabilities. While most previous research work on multi-robot cooperation mainly concentrates on loosely-coupled multi-robot tasks, a more challenging problem is to also address tightly-coupled multi- robot tasks involving close robot interactions, which often require capability sharing. Three related topics towards addressing these tasks are discussed, as follows:

Forming coalitions, which determines how robots should form into subgroups (i.e., coalitions) to address individual tasks. To achieve system autonomy, the ability to identify the feasibility of potential solutions is critical for forming coalitions. A general IQ-ASyMTRe architecture, which is formally proven to be sound and complete in this research, is introduced to incorporate this capability based on the ASyMTRe architecture.

Executing coalitions, which coordinates different robots within the same coalition during physical execution to accomplish individual tasks. For executing coalitions, the IQ-ASyMTRe+ approach is presented. An information quality measure is introduced to control the robots to maintain the required constraints for task execution in dynamic environment. Redundancies at sensory and computational levels are utilized to enable execution that is robust to internal and external influences.

Task allocation, which optimizes the overall performance of the system when multiple tasks need to be addressed. In this research, this problem is analyzed and the formulation is extended. A

new greedy heuristic is introduced, which considers inter-task resource constraints to approximate the influence between different assignments in task allocation.

Through combining the above approaches, a framework that achieves system autonomy can be created for addressing multi-robot tasks.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Gerkey [Gerkey and Mataric, 2004] categorizes robot problems based on the robots (single-task (ST) vs. multi-task (MT)), the tasks (single-robot (SR) and multi-robot (MR)) and the assignments (instantaneous (IA) and time-extended (TA)). In this dissertation, although the discussions are concentrated on addressing multi-robot tasks with single-task robots and instantaneous assignments, they are also immediately applicable to addressing single-robot tasks with single-task robots. The introduced approaches can also be used in problems involving time extended assignments. In this research, it is not assumed that individual robots have all the required capabilities, and hence robots need to form subgroups or coalitions in order to accomplish the tasks. When close coordinations between the robots are required, the tasks are referred to as tightly-coupled multi-robot tasks. Whether a task is loosely or tightly coupled is dependent on the task requirements, robot capabilities, and the current environmental situation. Since these factors can vary significantly across different applications, it is desirable to enable both loosely and tightly coupled tasks, in order to fully use the capabilities of the systems. In tightly-coupled multi-robot tasks, the robots are required to share their capabilities via constant information sharing. This research is aimed to provide a complete solution to achieve system autonomy for addressing various multi-robot tasks, whether loosely or tightly-coupled.

## 1.1 Forming coalitions

For accomplishing multi-robot tasks, robots are required to cooperate by forming subgroups (i.e., coalitions). An intuitive approach for reasoning about forming coalitions to address multi-robot tasks[1] is to divide them into subtasks or roles that individual robots can perform using domain knowledge. For example, in a robot insertion task [Sujan and Dubowsky, 2005], a supervisor robot provides visual information to guide the implementor robot to execute the insertion. An issue with this approach is that subtasks or roles have to be predefined, as it is not practical to define all of them for arbitrary tasks. It is desirable, and sometimes even required, for the reasoning to be dependent dynamically on the capabilities (whether sensory, motor, or computational) of the available robots, such that coalitions can be formed at a more fine-grained scale. Moreover, when individual robots do not have all the capabilities desired, they must reason autonomously to share them. This is desirable when it is either impractical or uneconomical to install the required sensors on all (potentially heterogeneous) robots, or when using sensors positioned on different robots is more suitable in the current situation. Forming coalitions based on subtasks or roles is often not informative enough for capability sharing.

The representation of robot capabilities clearly has a critical influence on the design of such desired systems. First, the capabilities should be defined independently of the robots and tasks so that the implemented systems are general and can be applied to various tasks. Modularity is also important to make the systems easily extendable to different problem domains requiring dissimilar capabilities, such as extending the robot insertion task to a robot box pushing task. Finally, it is necessary to define a uniform interface between these capabilities in order to autonomously reason about the required interactions among the robots and the environment.

General methods satisfying the independence and modularity requirements greatly increase system capability. Approaches that also enable autonomous reasoning with capability sharing have been shown in [Parker and Tang, 2006, Shiroma and Campos, 2009] to further improve the

[1] In this context, forming coalitions for individual multi-robot tasks is considered.

system flexibility in tightly-coupled multi-robot tasks. However, an important issue that remains unaddressed is that the satisfaction of the constraints (introduced by the interactions) are not considered; these constraints determine the feasibility of the potential coalitions. For example, in a navigation task, a robot without a localization capability may need help from another robot in order to navigate. This, in turn, requires the robots to interact in close proximity since the relative position between the robots has to be retrieved using sensors of limited range (e.g., cameras). A coalition with robots that are not in each other's sensor field of view (FOV) is infeasible. It is clear that the feasibility of the coalitions for execution is not only influenced by the capabilities of the robots, but also by the configurations of the robots and the environment. Previous approaches for forming coalitions do not consider this issue and thus cannot easily identify whether or not the formed coalitions are feasible for execution.

## 1.2 Executing coalitions

The second step to address a multi-robot task[2], is to address the problem of coalition execution. However, general methods for coalition execution do not yet exist, especially for tightly-coupled multi-robot tasks, in which close coordination with information sharing between the robots is required. The difficulty lies in the fact that the close robot coordination for task execution sets up interaction constraints among robots and the environment; these interaction constraints can in turn be expressed as sensor constraints that must be maintained by the robots during task execution. These constraints must be determined dynamically in the current situation. For example, in most cases, it is impractical for spatially distant robots to be assigned the same task when there are sensor constraints between them. In a navigation task, a robot without a localization capability can request another robot to constantly share its global position during execution; combining this global position with relative position between the robots can allow the first robot to calculate its own global position. However, the robots need to be in close proximity of each other for this to

[2]In this context, execution for individual tasks is considered.

work, since the relative position between the robots has to be retrieved using sensors with limited sensing ranges (e.g., fiducial sensors). To solve this problem in a general manner, a method is needed for modeling and dynamically identifying the required interactions in the multi-robot task. These interactions, in turn, determine the physical robot configurations required to satisfy the sensor constraints in the current situation.

To cope with dynamic and environmental influences while maintaining the required robot configurations, a measure is needed that assesses the utility of a multi-robot configuration for satisfying these sensor constraints, subject to these influences. Robots can use this measure to adjust their current configurations locally for maintaining these constraints when necessary. For cases when certain sensor constraints become unsatisfied, robots can search for alternative interactions to relax the unsatisfied constraints, without reallocating the task. This new approach can discover these new solutions using redundancies at the sensory and computational levels, thus providing more robustness during execution. Such a process is referred to as *constraint relaxation*, since it is the process of replacing the original sensor constraints with alternatives.

## 1.3 Task allocation

When there are multiple multi-robot tasks to be assigned, the optimization problem is referred to as the ST-MR-IA problem. The ST-MR-IA problem requires assigning a set of tasks to a set of robot coalitions, with the constraint that each robot and task can be assigned to no more than one coalition in the chosen assignments (i.e., coalition-task pairs). One effect of this constraint is that the capabilities on the robots are not sharable between different chosen assignments. The goal is to maximize the sum of the utilities of the chosen assignments.

The ST-MR-IA problem is closely related to the coalition formation problem in multi-agent systems. In the coalition formation problem, a set of agents replaces the set of robots and there is a function that maps a coalition of agents to a real nonnegative utility value [Sandholm et al., 1999]. The goal is to find a partition of this set of agents (i.e., referred to as a coalition structure

in [Sandholm et al., 1999]) to maximize the sum of the utilities of the coalitions in the partition. Compared to the coalition formation problem, the ST-MR-IA problem is slightly different in that it also requires a notion of task and incorporates an extra constraint on the tasks. Furthermore, it is not necessary to assign every robot to some task, since there may not be suitable or sufficient tasks to be assigned[3]. Moreover, the utilities of assignments in the ST-MR-IA problem are not only dependent on the coalitions, but also on the tasks. As a result, different assignments with the same coalition can have different utilities. Due to these differences, most algorithms from the agent-based coalition formation problem cannot be directly applied to the ST-MR-IA problem.

The ST-MR-IA problem can be easily shown to be NP-hard via a reduction from the coalition formation problem[4], which is known to be NP-hard [Sandholm et al., 1999]. In [Gerkey and Mataric, 2004], the ST-MR-IA problem is further shown to be strongly NP-hard [Garey and Johnson, 1978] by a similar reduction from the set partitioning problem. As a result, fully polynomial approximation algorithms for ST-MR-IA are unlikely to exist (unless $P \equiv NP$). Due to this complexity, few approximation algorithms with good solution guarantees have been provided. In this research, a new heuristic is presented that considers inter-task resource constraints in task allocation. This heuristic takes into account the influence between different assignments while still maintaining polynomial running time. A formal analysis is provided for this new heuristic, which reveals that the solution quality is bounded by two different factors. Algorithms based on this heuristic are easy to implement and simulation results show that they indeed improve the performance.

Although scheduling is not addressed in ST-MR-IA[5], for more complicated situations involving task dependencies, the formulation of ST-MR-IA is insufficient. For example, in a disaster response scenario [Jones et al., 2011], in order for truck agents to address fires in buildings, bulldozer robots must be assigned along with the truck agents to clear city roads leading to these buildings that are blocked by impassable debris. The task to clear city roads makes the task of addressing fires

[3]In ST-MR-IA, it is also not necessary for every task to be assigned to some robot(s), since there may not be suitable or sufficient robots to assign.

[4]The reduction creates a special task for each coalition and assigns the utility value of the assignment according to the utility function, while assigning the utility values of assignments with other coalitions for the task as zeros.

[5]Scheduling is typically considered in time extended assignment (TA).

possible. On the other hand, when there are alternative blocked roads that lead to the same buildings, the bulldozer robots only need to clear one of them. In this situation, the task to clear one road makes the other alternatives unnecessary (so that more bulldozer robots remain available for other tasks). It is clear that disregarding these task dependencies can significantly reduce the efficiency of the overall system. In this research, the formulation of the ST-MR-IA problem is extended to incorporate general task dependencies and provide an analysis of the complexity for the extended formulation. An algorithm that utilizes the discussed methods for ST-MR-IA is provided to address this extended formulation of the problem.

## 1.4 Task allocation with executable coalitions

The ST-MR-IA problem is very difficult to solve for several reasons. For one, the number of possible coalitions grows exponentially with the number of robots. Furthermore, given a set of possible coalitions ($C$), one needs to check every possible set of assignments of coalitions to tasks ($T$) in order to determine the optimal solution; the number of such sets is $O(|T|^{|C|})$. As the number of coalitions increases, the problem quickly becomes intractable. Even when the size of the coalitions is restricted as in [Shehory and Kraus, 1998], finding the optimal solution is still of high-order polynomial computational complexity.

Although efficient approximation algorithms or heuristics exist for addressing the ST-MR-IA problem, to enable these methods to run more efficiently, it is desirable to reduce the number of possible coalitions. Given a task, the most obvious way to reduce the number of coalitions is to consider the ones that satisfy the capability requirement of the task. However, the number of *feasible*[6] coalitions can still be large. For environments that are not guaranteed to be non-super-additive, the situation becomes even worse.

[6]For discussions of task allocation in this research, feasible coalitions represent the coalitions that satisfy the task requirements, which differ from the definition in previous discussions of forming and executing coalitions (in which they refer to coalitions that are *feasible for execution*). *Executable* coalitions are used in the latter case when there is ambiguity.

One observation is that, often, not all feasible coalitions are necessarily executable by the robots. This is due to the fact that a large portion of feasible coalitions may not be in an executable state (i.e., certain preconditions, or the required information of the tasks is not satisfied), such that the robots do not know how to execute them. These coalitions can be ignored for task allocation until ways to satisfy their preconditions are found and evaluated. Meanwhile, if the related tasks can be accomplished by executable (alternative) coalitions with reasonable costs, the satisfaction of the preconditions to enable these coalitions is likely to be unnecessary. To achieve this, the approach for forming coalitions in this research is used, which returns solutions in which the required interactions are satisfied.

For task allocation, each executable coalition is associated with a cost measure, which is evaluated to approximate the overall costs incurred for contributing to a task. Costs of the allocated capabilities (e.g., sensors), communication, and coordination between robots are all considered. A reward is also associated with each task and there are dependencies (e.g., precedence orders) between different tasks that must be satisfied. Meanwhile, for tasks with no executable coalitions, instead of only directly planning on the unsatisfied preconditions[7], the reasoning process for forming coalition is also used to autonomously decompose them into satisfiable components and create *partial order plans* [Russell and Norvig, 2003] to satisfy them accordingly (implemented using task dependencies).

## 1.5 Contributions

This dissertation makes contributions in several areas for addressing multi-robot tasks. The specific contributions for these areas are listed below. In the IQ-ASyMTRe architecture [Zhang and Parker, 2010b, Zhang and Parker, 2012c] for forming coalitions, we:

[7] Directly planning on these unsatisfied preconditions is only helpful if specific behaviors are implemented to satisfy these preconditions.

1. *Associate referents with information types.* This association completes the definition of information type, which is essential for identifying coalitions for which the required interactions are satisfied.

2. *Introduce information conversions.* Such an approach introduces a structured method for autonomously reasoning about the conversions between different information systems and hence avoids application specific software design.

In the IQ-ASyMTRe$^+$ approach [Zhang and Parker, 2010a, Zhang and Parker, 2011, Zhang and Parker, 2012b] for executing coalitions, we:

1. *Introduce environment sampling.* This method introduces a flexible way to incorporate environmental influence.

2. *Introduce and incorporate the information quality measure.* During execution, this measure is used to satisfy and maintain sensor constraints in tightly-coupled multi-robot tasks.

3. *Apply IQ-ASyMTRe during coalition execution for constraint relaxation.* Flexible and robust coalition execution to internal and external influences is achieved by using redundancies at the sensory and computational levels.

For task allocation [Zhang and Parker, 2012a], we:

1. *Provide a formal analysis on two natural heuristics.* This analysis shows that the natural heuristics can produce poor solutions when inter-task resource constraints are present.

2. *Introduce a new approximation algorithm and provide efficient ways to implement it.* A new approximation algorithm with solution quality bounded by two different factors is presented, which has been shown to improve the average performance.

3. *Extend the formulation of the problem and provide results on the hardness of this new formulation.* This analysis proves that this extended problem is NP-hard to approximate.

Through combining these approaches [Zhang and Parker, 2012d], we:

1. *Introduce task allocation with executable coalitions.* This method can greatly reduce the complexity for task allocation.

2. *Present a method to satisfy preconditions for tasks that have no executable coalitions, by breaking unsatisfied preconditions into satisfiable components.* Such an approach enables robots to autonomously create task plans to accomplish the tasks.

3. *Provide a framework to combine previous aspects to achieve system autonomy for addressing multi-robot tasks.* An initial framework is presented, which combines the previous important aspects to achieve multi-robot tasks.

The combined framework enables autonomous task planning and execution that exploits the capability of the current system for various tasks, which represents an important step towards achieving full autonomy in distributed robot systems. Given task specifications, robots using this framework can reason about coalition solutions dynamically based on the current configuration of robot teammates and the environment. The task allocation method can then be used to optimize the performance. During execution, robots can autonomously share capabilities in the formed coalitions and can adapt to dynamic and environmental changes in a flexible way.

The remainder of this dissertation is organized as follows. Related works are reviewed in Chapter 2. Chapter 3 presents the IQ-ASyMTRe architecture for forming coalitions. Chapter 4 presents the IQ-ASyMTRe$^+$ approach for executing coalitions, while task allocation methods are discussed in Chapter 5. A brief discussion of task allocation with executable coalitions is provided in Chapter 6 to demonstrate how these different aspects can be combined to achieve system autonomy for addressing multi-robot tasks. Results for these approaches are presented jointly in Chapter 7. Finally, conclusions are made in Chapter 8.

# Chapter 2

# Literature review

Research related to this dissertation includes the topics of multi-robot cooperation, task execution and coalition formation in multi-agent systems. This chapter provides a review of some of the most relevant work. First, existing architectures for forming coalitions are reviewed in Section 2.1. Then in Section 2.2, related approaches on executing coalitions are discussed. A review of algorithms for task allocation is provided in Section 2.3, while related work on task allocation with executable coalitions is discussed in Section 2.4.

## 2.1 Architectures for coalition formation

An extensive amount of work [Botelho and Alami, 1999, Dias and Stentz, 2000, Fua and Ge, 2005, Gerkey and Mataric, 2001, Werger and Mataric, 2000, Zlot and Stentz, 2005, Zlot et al., 2002, Parker, 1998, Gerkey and Mataric, 2000] has been proposed to address multi-robot cooperation with single-robot tasks, in which tasks are independent. Compared to this problem, addressing multi-robot tasks is considered to be more complex. Approaches that divide multi-robot tasks into subtasks or roles using domain knowledge [Das et al., 2002, Sujan and Dubowsky, 2005] effectively reduce them to single-robot tasks so that the previous work can easily apply. Variants of the Contract Net Protocol [Smith, 1980] are often used to coordinate the cooperation between multiple robots [Gerkey

and Mataric, 2001, Gerkey and Mataric, 2000]. However, these approaches can severely limit the capability of the systems when robots are required to share capabilities.

To reason about forming coalitions at a more fine-grained scale, researchers working on the task allocation problem often adopt a numeric representation [Shehory and Kraus, 1998, Shehory and Kraus, 1999, Klusch and Gerber, 2002, Sandholm et al., 1999, Sandholm and Lesser, 1995] for robot capabilities. The advantage is that this representation is general and more conveniently subject to theoretic analysis. The disadvantage, however, is that it does not facilitate capability sharing. Meanwhile, a behavior-based representation, such as schema theory [Lyons and Arbib, 1989, Arkin, 1989], is designed for modularity and provides an interface (i.e., inputs and outputs) for describing the interactions between different modules, although the unstructured interface is too generic to enable autonomous reasoning of potential coalition solutions. An advantage of this approach is that more complex behaviors can be achieved using the basic ones so that high-level decision-making architectures [Fikes and Nilsson, 1971, Parker, 1998, Saffiotti, 1997, Estlin et al., 2001] can be applied.

Although schema theory cannot be directly applied to facilitate autonomous reasoning, it is a promising method to build upon. In light of this, the ASyMTRe [Parker and Tang, 2006] architecture was introduced for addressing tightly-coupled multi-robot tasks; it is also the first architecture to enable autonomous capability sharing at the sensory and computational levels. For more structured inputs and outputs, ASyMTRe uses *information type*[1] as labels for attaching information with semantic meanings. Schemas (i.e., function modules) can only be activated when their input information types are satisfied, and can produce specified output information types. In this way, ASyMTRe is able to autonomously reason about the required interactions for potential coalition solutions based on how information should flow within the distributed systems to where it is required. Capability sharing is implicitly achieved through the communication of information. Unlike the

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

sensor fusion architectures [Murphy, 1998, Stroupe et al., 2001] that address the problem of combining sensory information from different robots, ASyMTRe is designed to model the information flow between very different modules (e.g., motors and sensors).

While other approaches for addressing tightly-coupled multi-robot tasks exist, some of them do not enable autonomous reasoning, due to the lack of a uniform interface between the capabilities. As a result, these approaches cannot be easily extended to various problem domains. For example, the work of [Kalra et al., 2005] introduces a market-based framework for tight coordination in a security sweep task domain. Passive coordination is used to quickly produce solutions for local robots, while active coordination is used to produce complex solutions via coordination between teammates. Other approaches that enable autonomous reasoning, however, do not facilitate capability sharing. Unlike in the multi-agent domain, sensory capabilities are located on different robots and cannot be easily transferred. Vig [Vig and Adams, 2006] first noted this, and addresses it by restricting coalitions to the ones that satisfy the location constraints of the sensors. A better solution is for the robots to autonomously share capabilities as enabled in [Parker and Tang, 2006, Shiroma and Campos, 2009].

Meanwhile, although application-specific methods [Parker et al., 2009, Spletzer and Taylor, 2002, Das et al., 2002] can often be used, such approaches do not generalize, even in the same task domain, to scenarios that require complex information sharing, due to the large number of possible interactions. Hence, a general architecture is required, and a schema-based design, coupled with a structured interface based on information types, satisfies the requirements for addressing tightly-coupled multi-robot tasks. However, while this approach enables autonomous reasoning about the required interactions for potential coalition solutions, it cannot determine which ones are feasible in the current situation, since the introduced constraints are not considered. The information type (for specifying the interactions) is statically defined with respect to the capabilities of the schemas and does not include information on how the coalition solutions are currently being instantiated.

## 2.2 Approaches for coalition execution

While many approaches have been provided for forming coalitions [Botelho and Alami, 1999, Dias and Stentz, 2000,Das et al., 2002,Fua and Ge, 2005,Gerkey and Mataric, 2001,Zlot and Stentz, 2005, Shehory and Kraus, 1998, Vig and Adams, 2006, Kalra et al., 2005, Parker and Tang, 2006, Zhang and Parker, 2010b], no general approaches have been provided to execute them. Coalition execution is especially important when addressing tightly-coupled multi-robot tasks, since such tasks require robots to interact in a constrained manner with each other, and/or with the environment. In particular, the required interactions introduce sensor constraints that must be maintained during execution. These sensor constraints in turn impose restrictions on the robot configurations during coalition execution. While previous work often uses application-specific methods [Howard et al., 2006, Sujan and Dubowsky, 2005] to maintain the required robot configurations, general methods for related problems can also be applied. One approach is to employ planning techniques [Ayanian and Kumar, 2010,Ogren and Leonard, 2003] to plan paths that ensure the maintenance of the robot configurations. The planned paths can then be used to feed back commands to control the robots. However, this approach is computationally expensive. Furthermore, dynamic influence is difficult to incorporate.

A more suitable approach is to use formation control techniques to maintain the required robot configurations. Formation control in dynamic environments has been classified into three categories based on the control strategy [Michaud et al., 2002]. The first category uses linear or non-linear control [Das et al., 2002, Desai et al., 2001, Antonelli and Chiaverini, 2006, De la cruz and Carelli, 2008,Monteiro and Bicho, 2010], in which dynamic and environmental influences are often incorporated by creating virtual robots on obstacle surfaces. However, these methods often do not consider sensor limitations (e.g., field of view) and assume visibility of at least the neighboring robots. Although a few methods in this category consider these limitations [Gustavi and Hu, 2005], the typical approach is to simply control sensor-constrained robots to be close enough to the sensor. The second category of methods uses a behavior-based approach [Balch and Arkin, 1998]. These methods

can often be implemented to rely only on local sensors [Fredslund and Mataric, 2002, Lemay et al., 2004, Ren and Sorensen, 2008, Barnes et al., 2009]; dynamic and environmental influences are often incorporated using potential fields [Fredslund and Mataric, 2002, Barnes et al., 2009]. The third category is a hybrid of the previous two categories. For methods in all three categories, sensors are not modeled to consider dynamic and environmental influences. As a result, these methods may lead to undesirable behaviors while maintaining the required sensor constraints for the task.

To address this issue, a measure is introduced to assess the utility of a multi-robot configuration for satisfying sensor constraints. This measure is based on the use of sensor models. Although other researchers have investigated similar measures, they do not explicitly consider dynamic and environmental influences. As a result, these previous measures can also lead to undesirable behaviors while maintaining sensor constraints. For example, in prior work for a target observation task [Stroupe and Balch, 2003], the robots compute a utility measure based on information gain to consider the observational contributions of teammates at each step; the robots then choose their motions to maximize this utility measure. However, dynamic and environmental influences are not considered. In [Spletzer and Taylor, 2002], a utility measure is computed by a function that is based on a multi-variant model (defined for multiple observing robots), given the current robot positions. The robots are controlled to maximize this utility measure, which can be defined differently for achieving various tasks. However, dynamic and environmental influences are only incorporated using potential fields. In this research, these influences are considered explicitly with an approximated geometric representation of the environment, which is then used by the sensor models to compute the desired measure.

Nevertheless, this desired measure cannot guarantee the maintenance of the required sensor constraints in all situations, due to unadaptable dynamic and environmental influences or sensor failures. When certain sensor constraints introduced by the required interactions for the task are no longer satisfied, the execution cannot continue. Previous methods often achieve robust solutions by using redundancies at the robot level [Parker, 1998, Fua and Ge, 2005, Michaud et al., 2002]. These methods require other available robots to be allocated, which may be inefficient or

impractical. For example, it may be expensive to bring new robot coalitions together. In this research, it is demonstrated how more robustness can be achieved by using redundancies at the sensory and computational levels via constraint relaxation. This relaxation process often results in significant changes of the robot configurations, similar to formation switching. In formation control, however, this process is often hard-coded or manually controlled [Fierro et al., 2001, Das et al., 2002, Desai et al., 2001]. In contrast, the new approach allows robots to flexibly change the required task interactions; this approach relaxes the unsatisfied constraints autonomously, based solely on whether the desired information flow can be maintained in the current situation.

## 2.3 Algorithms for task allocation

The coalition formation problem (similar to ST-MR-IA) has been studied extensively as *characteristic function games* in multi-agent systems (e.g., [Abdallah and Lesser, 2004, Rahwan et al., 2009, Sandholm et al., 1999]), which concentrate on generating optimal coalition structures. Sandholm et al. [Sandholm et al., 1999] show that for any algorithms to obtain solution guarantees, the search process is required to visit an exponential number of coalition structures in the number of agents. Sandholm et al. [Sandholm et al., 2002] also show that it is difficult to approximate the problem using techniques from combinatorial auctions. Nevertheless, researchers have proposed efficient algorithms for this problem. In [Abdallah and Lesser, 2004], a novel distributed algorithm is presented that returns a solution in polynomial time, given an underlying hierarchical organization. Reinforcement learning techniques are used to increase the solution quality as the agents gain more experience. In [Rahwan et al., 2009], an efficient anytime algorithm is provided that uses a novel representation of the search space to partition the solution space and remove unpromising sub-spaces. The branch-and-bound technique is then applied to reduce the search of the remaining sub-spaces. A similar problem is the set partitioning problem, for which many algorithms have been provided (e.g., [Atamturk et al., 1995, Hoffman and Padberg, 1993]). However, these discussed

approaches cannot be used to address the ST-MR-IA problem due to the fact that the notion of task is absent.

The problem of coalition formation for task allocation has been studied in [Dang and Jennings, 2006, Lau and Zhang, 2003, Service and Adams, 2011, Shehory and Kraus, 1998, Tosic and Agha, 2004]. Lau and Zhang [Lau and Zhang, 2003] have introduced a taxonomy for this problem based on three factors: demands, resources and profit objectives. They have investigated five distinct classes of the problem and have provided analyses and algorithms for each class. In their formulation, coalitions are allowed to overlap so that the same robots can potentially be assigned to multiple tasks. This approach assumes that the capabilities on the robots are sharable between different coalitions, which does not apply to multi-robot systems [Vig and Adams, 2007]. Note that since task locations are often geographically distant, physical robots cannot execute different tasks simultaneously. As a result, these algorithms are not suitable for addressing the ST-MR-IA problem. Dang and Jennings [Dang and Jennings, 2006] studied the coalition formation problem in a task-based setting. They have provided an anytime algorithm that has bounded solution quality with a minimal search. Their formulation of the problem is in fact more general than ST-MR-IA, since multiple tasks are allowed to be assigned to any coalitions. However, they did not study the influence of the size of the coalitions on the solution quality.

Meanwhile, the formulations of the problem studied in [Service and Adams, 2011, Shehory and Kraus, 1998, Tosic and Agha, 2004] match more with that of the ST-MR-IA problem. In [Tosic and Agha, 2004], a fully distributed algorithm is presented, in which a maximal clique approach is applied for ensuring the high degree of communication connectivity of the candidate coalitions, thus providing more robustness. Task allocation is then achieved by selecting from these candidate coalitions based on utilities. However, the algorithm does not provide any solution guarantees. Shehory and Kraus [Shehory and Kraus, 1998] have adapted a greedy heuristic [Chvatal, 1979] from the set covering problem to address the task allocation problem via coalition formation in both multi-agent and multi-robot systems. They have studied two cases of the problem with non-overlapping and overlapping coalitions, in which the non-overlapping case is almost identical to

the ST-MR-IA problem, except that a cost measure is used instead of a utility measure. A non-super-additive environment is assumed so that they can bound the size of the coalitions. With this assumption, they have shown that the greedy algorithm produces a solution that is within a logarithmic factor of the optimal.

However, in the most recent work of [Service and Adams, 2011], Service and Adams point out that changing the optimization problem from a cost (as in [Shehory and Kraus, 1998]) to a utility measure has a great impact on the performance of the algorithm. They have studied two models of the problem in which the resource model is exactly the ST-MR-IA problem. It is proven in [Service and Adams, 2011] that it is NP-hard to approximate the solution within $O(|T|^{1-\epsilon})$ without restricting the size of the coalitions. In addition, it is NP-hard to obtain an approximation ratio that is asymptotically no worse than $k/\log(k)$ when the maximum size of the coalitions is restricted to $k$, using the results reported in [Zuckerman, 2007]. Service and Adams have also provided a greedy heuristic and reported an approximation ratio of $\theta = k + 1$ in the worst case. This same heuristic is presented in this research as one of the natural heuristics (i.e., *MaxUtility*). Another natural heuristic is also analyzed and a new heuristic is presented.

In the research of multi-robot systems, the task allocation problem has also been studied extensively [Fanelli et al., 2006, Fua and Ge, 2005, Parker and Tang, 2006, Sariel, 2005, Vig and Adams, 2006, Zlot, 2006] (some of these are not necessarily restricted to the ST-MR-IA problem [Fanelli et al., 2006,Sariel, 2005,Zlot, 2006]). Some approaches are designed to achieve specific objectives or tasks [Fua and Ge, 2005,Sariel, 2005,Zlot, 2006]. In [Fua and Ge, 2005], a backoff adaptive scheme is employed to enable fault-tolerant task allocation with uncertain task specifications. In [Sariel, 2005], a framework for a cooperative exploration task in dynamic environments is proposed. During execution, costs are re-evaluted in the current situation based on a cost function and robots can dynamically change targets to minimize the total costs. In [Zlot, 2006], a way to generalize task descriptions as task trees is provided, which is implemented in a distributed solution for allocating complex tasks (i.e., involving task scheduling and decomposition) using a market-based approach.

For approaches that provide techniques to address the general optimization problem, anytime algorithms are applied in [Fanelli et al., 2006, Parker and Tang, 2006] and heuristics are used to return potentially good solutions first. In [Vig and Adams, 2006], Vig and Adams adapt the approach in [Shehory and Kraus, 1998] to multi-robot systems. To address the location constraints of capabilities (e.g., in a tracking task, a camera must be mounted on a robot that is also mobile), they have designed a process to check the satisfaction of these constraints and remove assignments that violate them. As an alternative, Vig and Adams have introduced the service model (also studied in [Service and Adams, 2011]) to avoid this process. However, such an approach requires the services to be predefined for various tasks, which can potentially be dependent on the capabilities of the robots that are unknown.

## 2.4 Task allocation with executable coalitions

To provide a solution that can combine the previous aspects to achieve multi-robot tasks, a layering technique that allows different task allocation methods to be easily incorporated is adopted. A similar approach is presented in [Tang and Parker, 2007] for the ASyMTRe architecture [Parker and Tang, 2006]. However, as discussed prior in this research, ASyMTRe suffers from several issues which are addressed by IQ-ASyMTRe to enable coalition execution. As a result, some techniques used in [Tang and Parker, 2007] to improve performance (e.g., considering robots having the same capabilities for a task to be equivalent) reduces system autonomy, as the influence of dynamic factors and environment settings for execution cannot be considered. To provide a new aspect, the issue with a large number of coalitions is addressed in this research by allocating tasks to executable coalitions.

Finally, the issue is addressed when no executable coalitions exist for tasks. One approach is to use AI planning techniques separately as in [Lundh et al., 2007]. However, this approach is only applicable for sequencing executable behaviors, with the previous behaviors satisfying preconditions

of the following ones. The method presented in this research is able to find ways to satisfy these preconditions even no task behaviors are directly provided to satisfy them.

# Chapter 3

# IQ-ASyMTRe for coalition formation

For forming coalitions, IQ-ASyMTRe [Zhang and Parker, 2010b, Zhang and Parker, 2012c] is built based on the ASyMTRe architecture [Parker and Tang, 2006], which is the first architecture that enables autonomous capability sharing. The reasons for choosing ASyMTRe are discussed and its advantages and limitations are investigated. IQ-ASyMTRe is introduced specifically to address several limitations of ASyMTRe for forming coalitions. To the best of our knowledge, IQ-ASyMTRe is the first attempt to form coalitions that are executable. Note that this approach can also be used in coalition execution to identify situations when the feasibility of the coalitions changes dynamically, so that the robots can adjust their behaviors. More detailed discussions on this aspect can be found in Chapter 4. After some background knowledge about ASyMTRe in Section 3.1, the new IQ-ASyMTRe architecture is explained in detail in Section 3.2. Simulations and experimental results are presented in Section 7.1, and conclusions are made in Chapter 8.

## 3.1 ASyMTRe

Based on schema theory [Lyons and Arbib, 1989, Arkin, 1989], the ASyMTRe architecture [Parker and Tang, 2006] defines basic building blocks of robot capabilities to be collections of environmental sensors (ESs), perceptual schemas (PSs), motor schemas (MSs), and communication schemas

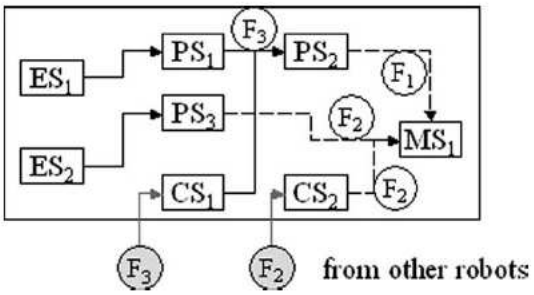



Figure 3.1: An example of how schemas are connected in ASyMTRe [Parker and Tang, 2006].

(CSs). A set of information types ($F$s) is introduced to label the inputs and outputs of schemas. Connections between schemas can be made when the output label of one matches the input label of another. To reason about coalition solutions, ASyMTRe searches through all possible ways to connect different schemas in order to activate the required MSs on the robots to achieve a task. The activation of each schema is associated with a cost. For task allocation, ASyMTRe uses an anytime algorithm, with heuristics for returning good solutions earlier. Figure 3.1 illustrates an example of schema connections. While solid lines represent an *AND* condition, dashed lines represent *OR*. For example, the figure shows that the requirement of information type $F_2$ can be provided by either $PS_3$ or $CS_2$. Refer to [Parker and Tang, 2006] for more details.

While ASyMTRe and other architectures (e.g., [Shiroma and Campos, 2009]) that use similar approaches have been shown to improve the flexibility of multi-robot systems, they all suffer from several issues. First of all, the definition of information type is not complete, which limits the referenced information to only be statically dependent on the capabilities of the schemas, instead of dynamically dependent on the actual information retrieved. This can cause problems when sensors capable of producing an information type cannot retrieve the required information in the current situation. For example, in a navigation task, a robot without a localization capability, and a robot with this capability, may not always be within each other's FOV. In such cases, these architectures may choose to form infeasible coalitions. To address this issue, schemas must be activated in the planning phase when necessary. The incorporation of such a process is missing in ASyMTRe

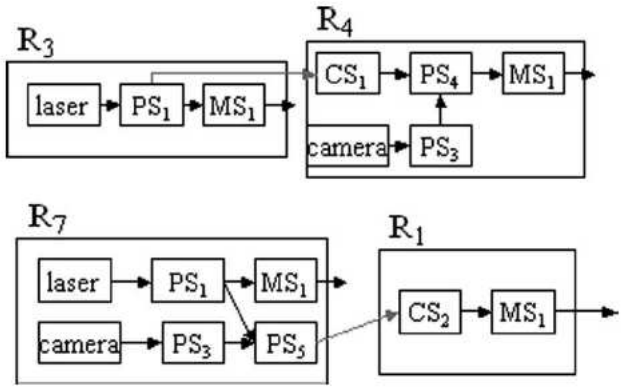



Figure 3.2: Retrieving the same information in different ways in ASyMTRe [Parker and Tang, 2006].

and other previous approaches for forming coalitions. Furthermore, perceptual schemas (or *actions* in [Shiroma and Campos, 2009]) can be dependent on how the input information is retrieved, which requires application-specific code to be designed. Figure 3.2 shows two scenarios for how robots without a localization capability ($R_4$ and $R_1$ in the figure) retrieve their global position information with help from another robot. In the top scenario, position information relative to $R_3$ is retrieved by $R_4$ using a camera, whereas in the bottom scenario, the relative information is retrieved by robot $R_7$ with a laser-based localization capability.

## 3.2 The IQ-ASyMTRe architecture

In this section, the representation of information is first extended to introduce a complete definition of information type and a new type of PS, called *Reduction PS*, to model the conversions between information. Such a representation is needed in order to dynamically reason about the feasibility of the coalitions. The new solution space and potential solutions are then presented and discussions are provided on how they relate to the coalition solution. Finally, the IQ-ASyMTRe algorithm for forming executable coalitions is presented and its properties are discussed and proven afterwards.

### 3.2.1 Information type and information instance

The incompleteness of *information type*, as originally defined in [Parker and Tang, 2006], is due to the fact that the relationships between entities[1] and information are not explicitly captured. Intuitively, information must be specified with a set of referents. For example, the information of $r_A$*'s global position* is not useful without specifying $r_A$. IQ-ASyMTRe uses both *information type* and *information instance* for a complete definition.

**Definition 3.2.1.** *Information Type – An information type in IQ-ASyMTRe is specified by a pair,* $(\mathcal{F}_i, N_i)$*, where* $\mathcal{F}_i$ *is a label for the semantic meaning of the information that defines the specific sensing or computational data of a schema or a sensor.* $\mathcal{F}_i$ *is consistent with the definition of information type in ASyMTRe, while* $N_i$ *is the number of referents that should be associated with* $\mathcal{F}_i$*.* □

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

**Definition 3.2.2.** *Information Instance – An information instance in IQ-ASyMTRe not only contains the semantic meaning expressed in its information type, but it also captures information about the related entities. An information instance of a particular information type,* $(\mathcal{F}_i, N_i)$*, can be represented as* $F_i(Ref_{1:N_i})$*, where* $Ref_j$ *is used to refer to the* $j$*th referent for the information instance.* □

Each referent, $Ref_j$, can be instantiated to a particular entity or remain uninstantiated for future instantiations. Fully instantiated information instances represent actual information that can be used. Partially instantiated information instances represent a class of information. For example,

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

$F_G(X)$ can be the global position information of any entity that $X$ is instantiated to. The use of information instance creates a complete reference of information. For example, after instantiating the two referents of an information instance of the relative position information type, ($\mathcal{F}_R$, 2), to robots $r_A$ and $r_B$ respectively, the reference of information $F_R(r_A, r_B)$ has a unique meaning (i.e., $r_A$'s position relative to $r_B$), no matter how the information is retrieved or used.

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

The concept of *generality* for information instances is introduced for later use. In the following discussions, for conciseness, referents for information instances are not shown unless necessary.

**Definition 3.2.3.** *Generality of Information Instance – For two information instances ($F^x$ and $F^y$) of the same information type ($\mathcal{F}$, $N$), $F^x$ is more general than $F^y$ (denoted by $F^x \succ F^y$), if and only if the following two statements are true:*

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

*(ii)* $\forall$ $Ref_k \in Ref_{1:N}$ *in* $F^x$ *that is instantiated, the corresponding referent* ($Ref_k$) *in* $F^y$ *is also instantiated.*

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

*(i)* $\forall$ $Ref_k \in Ref_{1:N}$ *in* $F^y$ *that is instantiated, the corresponding referent* ($Ref_k$) *in* $F^x$ *is also instantiated.*

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

□

Note that Definition 3.2.3 only specifies partial orders for information instances of the same information type in IQ-ASyMTRe, since two information instances of the same information type may or may not be comparable. For example, given that $F^x \not\succ F^y$ and $F^x \neq F^y$, it is not necessarily true that $F^y \succ F^x$.

### 3.2.2 Information conversion

The Reduction PS (denoted by *RPS* henceforth) is introduced in IQ-ASyMTRe for expressing information conversions. The idea is to provide a constructive way for reasoning about the relationships between different information systems, as first introduced in information invariants theory [Donald et al., 1997]. This capability is desirable for reasoning about forming coalitions, since potential coalition solutions represent ways to connect different components (i.e., schemas) to form equivalent information systems (i.e., to retrieve the required information). The benefit is that information conversions are general, so that no application-specific code needs to be designed for the reasoning process.

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

Information conversions expressed in IQ-ASyMTRe can be specified in the Backus-Naur Form (BNF) as follows.

Table 3.1: Examples of RPS's

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

**Definition 3.2.4.** *Information Conversion – Information conversions in IQ-ASyMTRe express relationships between composite information instances (abbreviated as comp inst in the BNF specifications).* □

A composite information instance is constructed from connected information instances using logic operators {*iAND*, *iOR*}, which are similar to {*AND*, *OR*} in propositional logic, except that *iOR* is more strict in that it specifies *either one but not both*. It is not difficult to conclude that the distributive property holds for {*iAND*, *iOR*} as with their counterparts in propositional logic. Information conversions convert the composite information instance on the left hand side to the one on the right:

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

The BNF of a composite information instance is given as follows, in which *information instance* is abbreviated as *info inst*. Since *iOR* operators on the right hand side are not well defined[2], the definitions for the two sides are different:

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

[2]For example, saying that the information instance of $A$ can either be converted to $B$ or $C$, but not both, turns the use of this information conversion into a random process.

$$< \textit{r-comp inst} > ::= (< \textit{r-comp inst} > \textit{iAND} < \textit{r-comp inst} >)$$

$$| < \textit{info inst} >$$

**Lemma 3.2.1.** *Information conversions in IQ-ASyMTRe can always be defined with only iAND operators connecting multiple information instances on the left hand side and a single converted information instance on the right.*

*Proof.* First, since information instances on the right are connected with only *iAND* operators, one can easily divide any information conversions in Definition 3.2.4 into multiple conversions with a single information instance on the right by creating a separate conversion for each information instance on the right with the left hand side unchanged. Afterwards, by using the distributive property, one can transform the composite information instances on the left into their respective disjunctive normal forms (DNFs). Finally, one simply needs to divide each of the transformed information conversions into multiple conversions by introducing a new conversion for each conjunctive clause on the left hand side. □

As an example, for an information conversion having the form, $(A\ iOR\ B)\ iAND\ C \Rightarrow D\ iAND\ E$, first, it can be divided into two information conversions for each information instance on the right hand side. The two conversions are $(A\ iOR\ B)\ iAND\ C \Rightarrow D$, and $(A\ iOR\ B)\ iAND\ C \Rightarrow E$. Next, apply the distributive rule to the first conversion to get: $(A\ iAND\ C)\ iOR\ (B\ iAND\ C) \Rightarrow D$. The second one can be transformed similarly. Finally, by introducing a conversion for each conjunctive clause, one has $A\ iAND\ C \Rightarrow D$, and $B\ iAND\ C \Rightarrow D$.

**Definition 3.2.5.** *IQ Information Conversion – Lemma 3.2.1 allows us to express any valid information conversion in the following form (called the 'IQ form' henceforth):*

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

$$< \textit{l-comp inst} > ::= (< \textit{l-comp inst} > \textit{iAND} < \textit{l-comp inst} >)$$

$$| < \textit{info inst} >$$

$$< \textit{r-comp inst} > ::= < \textit{info inst} >$$

□

The advantage of defining information conversions in the IQ form is that the reasoning process for creating the solution spaces is significantly simplified. In the following discussions, it is assumed that all information conversions are defined in the IQ form; the *iAND* operator is also denoted by '+', for conciseness. Table 3.1 shows several RPSs that can be used in IQ-ASyMTRe. They are general since the referents can be instantiated to different entities, as long as the same referent labels are instantiated to the same entities.

### 3.2.3 Solution space and potential solution

While potential coalition solutions or *potential solutions* represent the possible alternative ways that schemas can be connected, a *solution space* in IQ-ASyMTRe encodes all such potential solutions. Although the introduction of information instance and RPS significantly changes the solution space and potential solutions in IQ-ASyMTRe, the reasoning process for creating them remains similar to [Parker and Tang, 2006]. To create the solution space, the reasoning algorithm checks all schemas that can output the required information instances, and then checks recursively for the inputs of those schemas until the path either ends in a conflict with the *referent instantiation constraint*[3] or in a terminal state (i.e., CS or ES-EPS pair[4] that can be the source of the required

[3]The referent instantiation constraint requires the same labels to be instantiated to the same entities in the inputs and outputs of the same schemas. Validation of this constraint occurs in both of the algorithms presented in Section 3.2.7. In the algorithm for creating the solution space, it is used to remove invalid potential solutions, while it is used in the second algorithm to determine the feasibility of the coalitions.

[4]Sensory information instances are extracted from raw sensor data using PSs designed for the specific environmental sensors (ESs). PSs of this kind are denoted by *EPSs* in the following discussions.

information instance). Note that while the reasoning for creating the solution space is similar to STRIPS [Fikes and Nilsson, 1971] planning (considering sets of information instances as states and RPSs as reduction rules), IQ-ASyMTRe provides a more manageable reasoning system for the problems that this research is addressing, by restricting the forms of states and RPSs.

A solution space can be represented in a directed graph representation as an *and-or* tree (e.g., see Figure 3.3). Each node in the solution space represents a schema or an ES-EPS pair. Each edge represents an information instance that constantly flows from the output of one node into the input of another (except for the edges connecting ESs and EPSs). The root of the *and-or* tree, which specifies the required information instances, can either be a MS or CS (for providing information to other robots) and acts as a sink node that information flows into. As it is assumed that RPSs are defined in the IQ form, every node has one or more incoming edges and a single outgoing edge. Every leaf node (i.e., the information source) is either a CS, representing information communicated from others, or an ES-EPS pair, representing information retrieved using sensors. Nodes that are closer to the sink node (i.e., the root) are said to be downstream of the nodes further away on the same branching path[5], as information flows from the leaves to the root. Figure 3.3 shows a solution space for the cooperative robot navigation task. The *tOR* node is introduced to manage multiple options of connection. Another example involving different information types is provided in Figure 7.14 for the box pushing task presented in [Donald et al., 1997] using force feedback, infrared, bumper, and position sensors, which shows the applicability of IQ-ASyMTRe to varying task domains.

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

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

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

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

- *Generality Imposition* constraint – For each RPS, prohibit the use of the same RPS for converting information instances of the same type having less or equal generality upstream on the same branching path.

**Lemma 3.2.2.** *The Generality Imposition constraint ensures that solution spaces and potential solutions have finite graphical representations.*

*Proof.* The representation of a solution space can become infinite when loops occur, where information instances of the same type and having the same generality appear more than once on the same branching path of the solution space. The *Generality Imposition* constraint directly prohibits the occurrences of loops. Otherwise, as the numbers of RPSs, information types, referents associated with information types, and entities are finite, in the worst case, every branching path would terminate after information instances for all information types reach the most general forms (i.e., have no instantiated referents). Hence, solution spaces have finite representations. Furthermore, as potential solutions are trimmed solution spaces, the conclusion is straight-forward. □

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

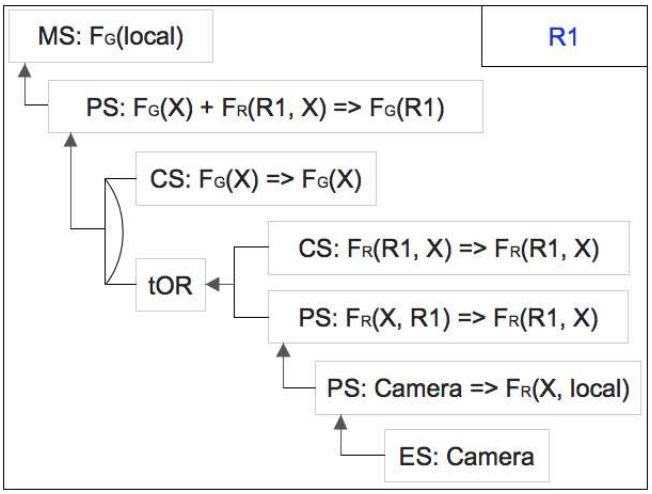



Figure 3.3: A solution space for a robot to obtain its global position with only a camera sensor. The referent *local* refers to the robot itself. This solution space encodes two solutions. One solution is to have another robot send over its global position (CS: $F_G(X) \Rightarrow F_G(X)$) and use the camera sensor to sense the relative position (EPS: Camera $\Rightarrow F_R(X, local)$). A 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.

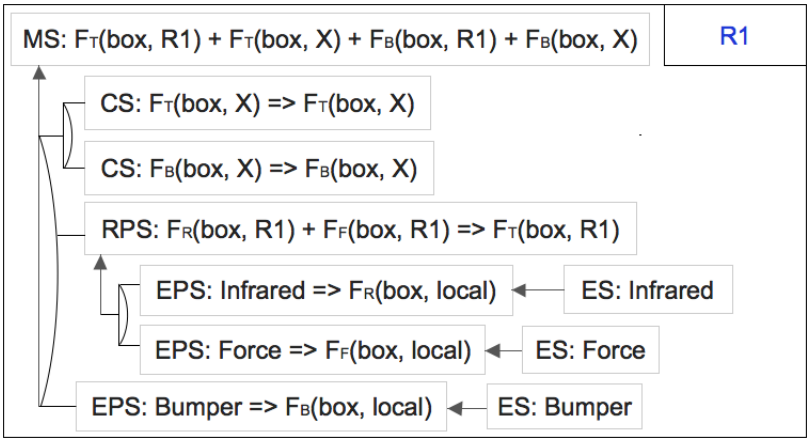



Figure 3.4: A solution space that represents one of the protocols presented in [Donald et al., 1997] for two robots to cooperatively push a box in a given direction. For the robots to coordinate actions, they must estimate the net torque on the box based on the torque information ($F_T$), and use the bumper information ($F_B$) to determine whether the robots are in contact with the box. The torque for each robot is computed from its exerted force on the box and its relative position to the box. Other protocols presented in [Donald et al., 1997] (e.g., using the positions of the robots relative to the box) can be similarly represented.

- *Localness in Reasoning* constraint – No schema connection except for CS should be created if none of the referents[6] for the information instance to be provided is instantiated to the local entity (except for those with all referents instantiated to non-robot entities).

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

- *External Communication* constraint – CSs are used only when some referents are not instantiated to the local entity.

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

### 3.2.4 Coalition and coalition solution

In IQ-ASyMTRe, potential solutions only specify how the *local robot*[7], $r_L$, directly interacts with others. To make this difference for later discussions, *local coalitions* are defined differently from traditional *coalitions* [Gerkey and Mataric, 2004].

**Definition 3.2.6.** *Local Coalition – A local coalition defined in IQ-ASyMTRe consists of $r_L$ and the robot teammates that directly feed information to $r_L$ for activating a MS or a CS.* □

To search for local coalitions, IQ-ASyMTRe checks potential solutions in the solution space in a non-descending order based on the costs. For each potential solution, IQ-ASyMTRe either activates the ES-EPS pairs to retrieve sensory information, or CSs to send out information requests, as specified in the potential solution. ES-EPS pairs are only activated temporarily to retrieve the

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

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

available sensory information and the information requests are sent only once (or a few times when communication links are unreliable). Robots receiving the information requests create a solution space for the required information and follow the same process. If the information is retrievable, these robots keep sending the information until the temporary activations of the schemas expire. IQ-ASyMTRe uses the collected information to instantiate the required information instances in the potential solution and perform validation of the referent instantiation constraints. The resulting local coalition is feasible only if all such constraints are satisifed. A chosen feasible local coalition can then be set up, by sending coalition requests to the relevant robot members. The instantiated potential solution is referred to as a *local coalition solution*, defined as follows.

**Definition 3.2.7.** *Local Coalition Solution – A local coalition solution (LCS) for a local coalition is the potential solution (selected by $r_L$) after the full instantiations of all information instances.* □

Note that a potential solution can be instantiated to different LCSs. The left part of Figure 3.5 shows the LCS for a valid instantiation of a potential solution in Figure 3.3. To define *coalitions*, note that robots in the local coalition may often be interacting with others in order to maintain the desired interactions. For example, in Figure 3.3, $F_G(X)$ may be retrieved by $X$ through help from another robot.

**Definition 3.2.8.** *Coalition – A coalition includes all robots in the local coalition and the robot teammates that indirectly support $r_L$.* □

A robot in the local coalition that uses CSs to directly feed information to $r_L$ must also be using a local coalition to retrieve the necessary information, which may involve yet other robots. Such local coalitions introduce coalitions (which could be of size 1) for these other robots in the local coalition for $r_L$. In other words, a coalition can recursively include multiple coalitions. Thus, the feasibility of a coalition can only be determined when the feasibility of all the required coalitions is determined; a coalition is set up only after all these required coalitions are set up. Note that while the cost of a local coalition is that of the corresponding potential solution, the coalition may incur more cost due to these additional required coalitions; this is referred to as the *coalition cost*.

**Definition 3.2.9.** *Coalition solution – The coalition solution for a coalition includes the LCS, as well as solutions for fulfilling the other required coalitions.* □

Coalition solutions also have *and-or* tree representations since they are created by connecting LCSs, which are *and-or* trees. One characteristic of the coalition solutions, which differs from LCSs, is that the leaves always represent the ultimate information sources, i.e., ES-EPS nodes. Note that dummy sensors can be created for providing any prior information when necessary. Figure 3.5 shows a valid coalition solution that instantiates a potential solution in Figure 3.3.

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

- *Distinct Requests* constraint – For any robot, prohibit the use of CS if the robot has already used CSs for requesting the same information instance.

**Lemma 3.2.3.** *Given the Generality Imposition and the Distinct Requests constraints, LCSs and coalition solutions have finite graphical representations.*

*Proof.* As LCSs are instantiations of potential solutions, given *Lemma 3.2.2*, the conclusion for LCSs is straight-forward. The representation of coalition solutions can become infinite when loops occur; this occurs only when one robot is requesting an information instance, which ultimately leads to itself requesting the same information again. The *Distinct Requests* constraint directly prevents loops from occurring. Hence, coalition solutions also have finite representations. □

Lemma 3.2.3 implies that any information request is populated only a finite number of times in the distributed system, although the information requested during the population may not necessarily be the same as in the initial request.

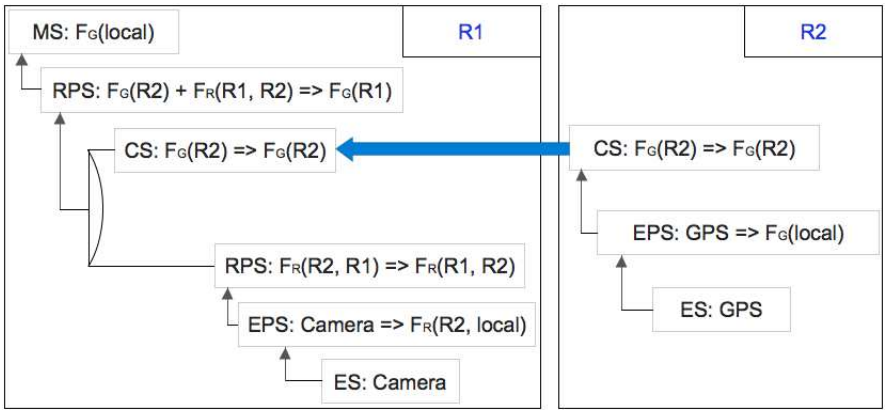



Figure 3.5: A possible coalition solution for a potential solution in Figure 3.3, in which the left part corresponds to the LCS.

### 3.2.5 The completeness of solution space

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

**Lemma 3.2.4.** *The combination of the following four constraints does not influence the completeness of the solution space for the distributed system:*

*(i) The Generality Imposition (GI) constraint;*

*(ii) The Localness in Reasoning (LR) constraint;*

*(iii) The External Communication (EC) constraint.*

*(iv) The Distinct Requests (DR) constraint;*

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

Without any constraints, the simplest solution is to have sensory information instances from other robots sent directly to $r_L$ (i.e., where the root is located) and move RPSs for processing them

to $r_L$. Since no constraints are imposed here, self-communicating CSs on $r_L$ are unnecessary and can be easily removed. Note that all information instances are fully instantiated in the solution just created. The goal is to show how this simple solution can be reconstructed equivalently (via pruning and grafting) on the robots so that the potential solutions before the instantiations satisfy all constraints.

Start thinking in a reverse order; that is, given the created simple coalition solution, what could the potential solutions have looked like? First of all, note that potential solutions are LCSs before the instantiations and the LCSs are part of the coalition solutions. Hence, any referent that is not instantiated statically (i.e., referents with instantiations specified a priori by the task or the EPSs) would not be instantiated in the potential solutions. After removing all non-static instantiations, it is shown next how to reconstruct the uninstantiated coalition solution to resolve all possible violations.

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

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

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

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

(c) In the case of a violation of the *GI* constraint, one knows that $\exists u \in$ *Down(v)*, for which $F_u \succ F_v$ or $F_u = F_v$. Furthermore, given the search order, it is also known that *Subtree(v)* does not contain any violations on $r_L$.

c.1) If there exists a referent in $F_v$ that is instantiated to a robot entity in the coalition, denoted by $r_E$ ($r_E \neq r_L$), *Subtree(v)* can be trimmed off and replaced with CS($F_v$). The creation of the CS does not violate any constraints; furthermore, *Subtree(v)* can be moved to $r_E$ for reasoning about $F_v$ without incurring any violations, by using the following process:

For leaf nodes that are ES-EPS nodes in *Subtree(v)*, they can be replaced with CSs after moving *Subtree(v)* to $r_E$ for requesting the sensory information instances from $r_L$. For the rest of the leaf nodes (CSs), if the transferred information is not from $r_E$, there is no need to do anything; otherwise, the process is more complicated. In the following, denote the information transferred from $r_E$ by $F_E$ and the node that is immediately upstream of CS($F_E$) on $r_E$ by $E$. If $F_E$ has more than one referent and one of them is instantiated to $r_E$, there is no need to do anything. Otherwise, there are two cases:

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

If $F_v$ has only one referent, following the same referent inheritance property, $F_u$, too, has a referent statically instantiated to $r_E$. According to the definition of the GI constraint, it must be that $F_v$ and $F_u$ represent the same information. As a result, one can move *Subtree(v)* back to $r_L$ to replace *Subtree(u)*. Otherwise, $F_v$ has more than one referent: if $E$ is an ES-EPS node, one can remove the leaf node CS($F_E$) from *Subtree(v)* and concatenate it with *Subtree(E)* on $r_E$; else, it holds that *Subtree(E)* must have violated the *LR* constraint before being moved from $r_L$ to $r_E$, since it clearly did not violate the *EC* constraint. Furthermore, it must not have violated the *GI* constraint, since otherwise *Subtree(E)* should have been used to replace the node in violation on $r_L$ (assuming that the resolution of the *GI* constraint takes precedence over that of the *LR* constraint). In such a case, one can remove the leaf node CS($F_E$) from *Subtree(v)* and concatenate it with *Subtree(E)* without a problem.

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

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

(d) In the case of a violation of the $LR$ constraint, it holds that none of the referents of $F_v$ is statically instantiated to $r_L$. A similar process as in (c) can be applied, given that information types are defined to have distinct semantic meanings.

Iterate the above process until all violations are resolved. On termination, an equivalent coalition solution has been created, in which the potential solutions before the instantiations satisfy all of our constraints. Hence, the conclusion holds. □

An example scenario for the navigation task is given in Figure 3.6(a) in which four robots must activate the same MS to go to the same global position. The robots are positioned in a column formation, as shown in the figure. The simple coalition solution for the example scenario is shown in Figure 3.6(b) for the bottom robot. Figure 3.6(c) shows the LCS before instantiations. Note a violation of the $GI$ constraint from two information instances is shown in red. Figure 3.6(d) shows the uninstantiated LCS after the resolution, with the red block in Figure 3.6(c) replaced by the blue block in Figure 3.6(d). However, a violation of the $LR$ constraint is still present in Figure 3.6(e) due to the information instance shown in red. Figure 3.6(f) shows the potential solution after resolving all violations and Figure 3.6(g) shows a possible coalition solution.

### 3.2.6 Forming executable coalitions

For searching and setting up coalitions, is is assumed that all robots are always within communication range, such that every robot can communicate with any other robot when necessary. Given a MS to activate, the robot first creates a solution space and searches for coalitions as discussed. Coalitions are feasible when all the required coalitions are feasible, and hence can be executed. No coalitions are set up in this phase. Among all executable coalitions found so far (after the given search time has elapsed), IQ-ASyMTRe chooses the one with the least coalition cost to set up. The coalition is only set up when all the required coalitions are also set up. For setting up coalitions, the same request-and-wait negotiation protocol is used as used in the distributed version of ASyMTRe [Tang and Parker, 2005].

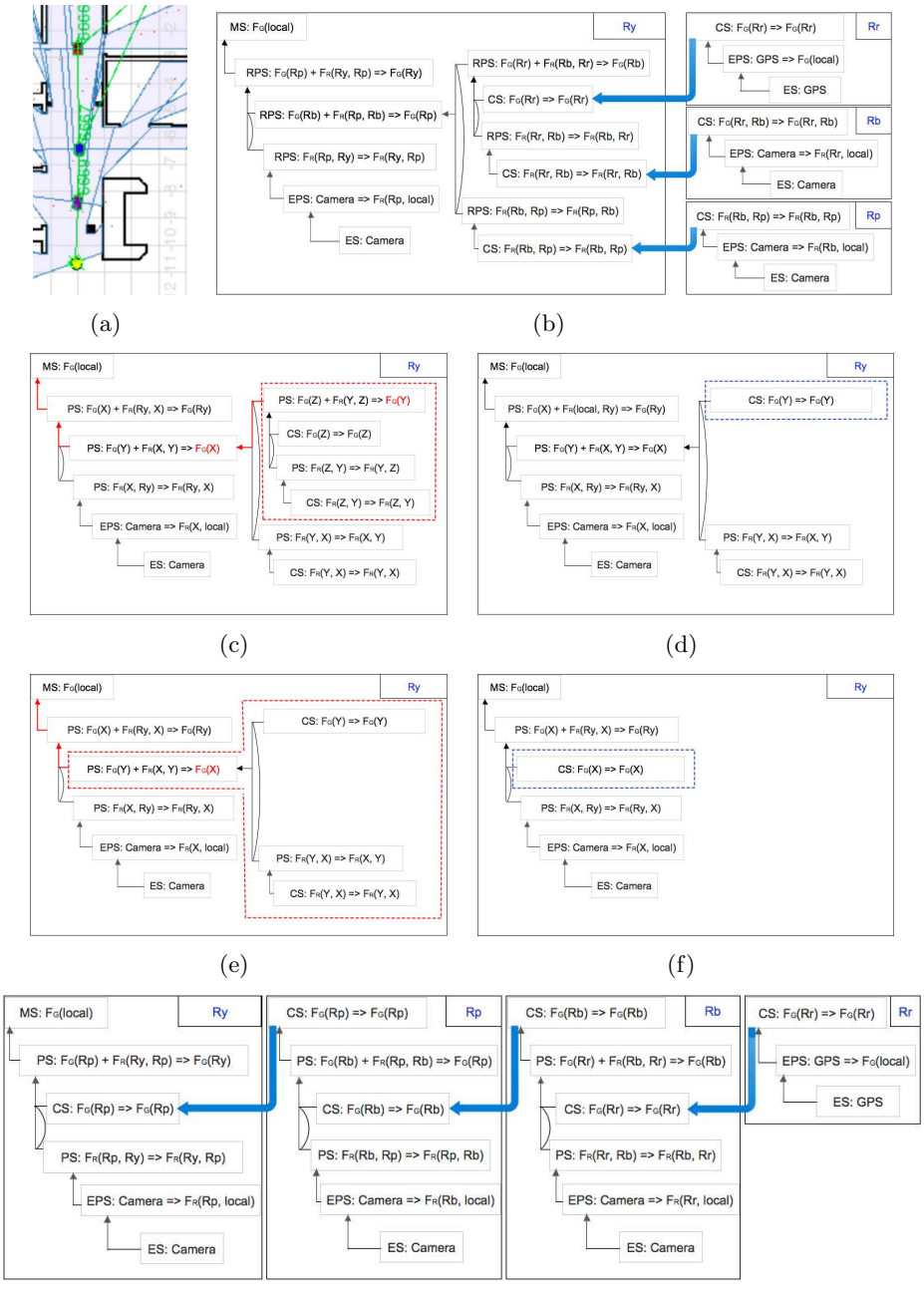



Figure 3.6: (a) A scenario for the navigation task in which only the red robot (at the top) has a GPS for global positioning. The other robots only have a camera for relative positioning, and each can only see the robot immediately in front of it, since they are in a column formation. (b) The simple coalition solution for the scenario in (a). (c) The LCS before instantiations. (d) After resolving the violation of the *GI* constraint. (e) A violation of the *LR* constraint. (f) After resolving the violation of the *LR* constraint. (g) A possible coalition solution after applying the potential solution in Figure 3.6(f).

### 3.2.7 The algorithms and properties

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

**Algorithm 1** A recursive algorithm for the solution space

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

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

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

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

**Theorem 3.2.5.** *If the set of RPSs expresses all valid information conversions, given sufficient search time, the IQ-ASyMTRe algorithm is complete.*

*Proof.* If the set of RPSs expresses all valid information conversions for a given application, then since Algorithm 1 explores all possible ways to retrieve information in a recursive manner, all potential solutions satisfying the constraints in Lemma 3.2.4 will be checked. Given that these constraints do not influence the completeness of the solution space, the IQ-ASyMTRe algorithm is complete given sufficient search time. □

**Theorem 3.2.6.** *The IQ-ASyMTRe algorithm is sound.*

*Proof.* First, since the inputs and outputs of connections are always required to have matching information types, solution spaces are created using only valid schema connections, as are the extracted potential solutions. Furthermore, since connections are made until the leaf nodes, which are always information sources, the required constant information flow can be maintained. For potential solutions, since one (and only one) branch remains at each *tOR* node, all inputs of the downstream nodes are still satisfied. Hence, all necessary connections are present in the potential solutions to maintain the information flow. Thus, the IQ-ASyMTRe algorithm is sound. □

### 3.2.8 Complexity analysis

The computational and space complexity of Algorithm 1 is linear in the number of input information instances to be reasoned about. For a single information instance, given the constraints in Lemma 3.2.4, it is not difficult to conclude that the worst case complexity of both computational and space complexity[8] is $O(N_t 2^{N_r} {N_c}^{N_t 2^{N_r}})$, given:

- $N_c$: the maximum number of RPSs producing the same information type.
- $N_t$: the number of information types related to the information instance to be reasoned about.

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

- $N_r$: the maximum number of referents associated with information instances for all related information types.

Given a specific domain, these numbers are fixed, so that the size of the solution space for each different information instance is bounded by $O({N_c}^{{N_c}^{N_t 2^{N_r}}})$. Meanwhile, the size grows exponentially with the number of information instances to be reasoned about. However, most practical problems are still small enough to be computed in reasonable time in practice. For applications in which the size of the solution space is large, the method in [Zhang and Parker, 2011] can be applied to achieve online performance by significantly reducing the size of the search spaces for certain problem instances.

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

# Chapter 4

# IQ-ASyMTRe$^+$ for coalition execution

In this chapter, the previous method of IQ-ASyMTRe for forming coalitions is used to model and provide information of the interactions among the robots and the environment. When dynamic factors and environment settings do not influence the task execution, robots use a localized formation control method to maintain the required robot configuration to satisfy the sensor constraints; otherwise, robots use an introduced measure to improve the utilities of these sensor constraints. Finally, the constraint relaxation process for flexible execution is presented, which is robust to internal and external influences. The new IQ-ASyMTRe$^+$ architecture [Zhang and Parker, 2010a, Zhang and Parker, 2011, Zhang and Parker, 2012b] for coalition execution is discussed in Section 4.1, with the key algorithms in Section 4.2. Simulations and experimental results are presented in Section 7.2.

## 4.1 The IQ-ASyMTRe$^+$ approach

The interactions among robots and the environment required for the task are modeled uniformly in IQ-ASyMTRe as information flows: CSs represent constant communication of information between robots; certain ES-EPS pairs (i.e., a sensor with the sensory data processing PS) represent the interactions that introduce sensor constraints, which must be satisfied to maintain these interactions (e.g., using fiducial sensors to retrieve the relative position). In IQ-ASyMTRe, these interactions

are dynamically determined by the capabilities and the current configurations (e.g., positions) of the robots, and the environmental situation. Presuming that inter-robot communication is not an issue, robots need to maintain the introduced sensor constraints during execution.

For forming coalitions, IQ-ASyMTRe ensures that the sensor constraints are satisfied in the initial robot configurations for the returned coalitions, by taking advantage of the complete reference of information (see [Zhang and Parker, 2010b] for details). During execution, the restrictions on the robot configurations imposed by these sensor constraints must be maintained. For example, in the navigation task, the sensor constraint is for the robots to keep each other in sight of the fiducial sensors; in a formation task that requires robots (with a ring of bumper sensors) to be in contact with others, the sensor constraint is for the robots to keep the bumper sensors activated. The required robot configurations for satisfying the sensor constraints (embedded in the initial configurations) can be considered as a formation constraint. Unlike a rigid formation, robots have some flexibility to adjust their configurations as long as the sensor constraints are maintained. In the navigation task, robots can adjust their relative positions, given that one robot remains in the other's sight. This gives robots more flexibility.

Let us denote the robot configurations for a robot coalition $R_j$ as $\mathbf{P}_j$ (referred to as a *coalition configuration*). Techniques introduced in this research can be summarized as follows:

---

Given initial $\mathbf{P}_j$.
**while** task not accomplished **do**
  **if** dynamic factors and environment settings do not influence the sensor constraints **then**
    Maintain formation specified by $\mathbf{P}_j$. {Section 4.1.1}
  **else**
    Find and switch to $\mathbf{P}'_j$ that satisfies the sensor constraints; set $\mathbf{P}_j$ to be $\mathbf{P}'_j$. {Sections 4.1.2 to 4.1.6}
  **end if**
**end while**

---

### 4.1.1 Maintaining Sensor Constraints using Formation Control

To accomplish a given task, a set of MSs, $\mathcal{M} = \{\text{MS}_1, \text{MS}_2, ...\}$, must be activated on different robots. In a tightly-coupled multi-robot task, each of these MSs may require information that must be cooperatively obtained by a coalition of robots; for each coalition $R_j$ (for $\text{MS}_j$), only one robot (denoted by $r_j^L \in R_j$) needs to execute $\text{MS}_j$, while others provide help by sharing necessary information. Also, for each $R_j$, the required interactions for this coalition introduce a set of sensor constraints, $\mathcal{C}_j = \{C_{j1}, C_{j2}, ...\}$, which must be satisfied during coalition execution. The subset of robots in $R_j$ that are involved in $C_{jk}$ is denoted by $R_{jk}$, and the robot with the sensor that introduces $C_{jk}$ is denoted by $r_{jk}^S$. Only one constrained sensor is associated with each $C_{jk}$.

When dynamic factors and environment settings do not influence the sensor constraints, the required coalition configuration for satisfying these constraints is maintained by using a localized formation control method, similar to [Fredslund and Mataric, 2002] but without assuming that cameras can pan. Each $R_{jk}$ acts locally to maintain the respective sub-configuration $\mathbf{P}_{jk}$ (a multi-robot configuration for satisfying sensor constraint $C_{jk}$) by using the related sensory information. Denote this information by $F_{jk}(Ref_{1:N_{jk}})$, in which $(F_{jk}, N_{jk})$ represents the information type. A special MS, denoted by $\text{MS}^\Delta$, implements this formation control for individual robots given such information. When all $R_{jk}$ are successful in maintaining the respective sub-configurations, the sensor constraints for maintaining the required interactions for coalition execution are satisfied.

For each coalition $R_j$, this process can be implemented using Algorithm 3. The algorithm first activates $\text{MS}_j$ on $r_j^L$. It then activates $\text{MS}^\Delta$ on robots that appear together with (or are linked to) $r_j^L$ in any sensor constraints in $\mathcal{C}_j$. The same process is performed on the robots with a newly activated MS, until all such robots are managed. For robots involved in any sensor constraints in $\mathcal{C}_j$ that are not directly or indirectly linked to $r_j^L$ (denote these robots by $R_j^N$), $\text{MS}^\Delta$ only needs to be activated when one of the associated referents in the related sensory information is instantiated

to a non-robot mobile entity (e.g., a moving target).[1] This algorithm is executed in IQ-ASyMTRe$^+$ for each coalition in a distributed manner, which runs in $O(|R_j|^2|\mathcal{C}_j|)$.

**Algorithm 3** Maintain Constraints with Formation Control

```
Create a queue Q and add r_j^L to Q.
Activate MS_j on r_j^L.
while Q is not empty do
  Remove the first element, r, from Q.
  for all sensor constraints in C_j, denoted by C_jk do
    if r ∈ R_jk then
      for all r' ∈ R_jk (r' ≠ r) do
        if r' does not have any activated MS then
          Have r_jk^S send F_jk(Ref_1:N_jk) to r' if r' cannot retrieve F_jk(Ref_1:N_jk).
          Activate MS^Δ on r' and add r' to Q.
        end if
      end for
    end if
  end for
end while
Activate MS^Δ on R_j^N if one referent of the related information instance is instantiated to a
non-robot mobile entity.
```

**Proposition 4.1.1.** *For each coalition $R_j$, given that all sensor constraints are initially satisfied, Algorithm 3 activates MS$^\Delta$ on all necessary robots to maintain the sensor constraints for activating MS$_j$.*

*Proof.* For any $C_{jk}$ that is associated with $R_j$: 1) If there exists a referent in $F_{jk}(Ref_{1:N_{jk}})$ that is instantiated to a robot or a non-robot mobile entity, the algorithm activates MS$^\Delta$ on all robots in $R_{jk}$ (except for $r_j^L$) to maintain $\mathbf{P}_{jk}$. In such cases, since there is at most one robot (i.e., $r_j^L$) in $C_{jk}$ that does not have MS$^\Delta$ activated or at most one non-robot mobile entity, no conflicts in motion are incurred. When any robot in $R_{jk}$ is involved in another sensor constraint $C_{jk'}$ $(k \neq k')$, since $\mathbf{P}_{jk}$ and $\mathbf{P}_{jk'}$ are consistent initially (both are a sub-configuration of $\mathbf{P}_j$) and are maintained, they

[1] One note is that since the motions of non-robot mobile entities are not always predictable, it is required for robots (including $r_j^L$) in any sensor constraints to only be (directly or indirectly) linked to other robots (assuming all robots are mobile) if they are linked to $r_j^L$, or to only one entity when this entity is a non-robot mobile entity. For more parallelism, exceptions may be considered in an application-specific manner.

must also be consistent during execution. Hence $C_{jk}$ is maintained. 2) Otherwise, $C_{jk}$ is trivially maintained. □

When two different coalitions $R_j$ and $R_{j'}$ in the given task must share a specific robot that is directly or indirectly linked to $r_j^L$ and $r_{j'}^L$ respectively, unless the current configurations of $r_j^L$ and $r_{j'}^L$ are maintained during execution, these tasks must be considered as two separate tasks. (Note that separate tasks can be handled by applying IQ-ASyMTRe$^+$ on each one sequentially.) For example, in a navigation task with two MSs to be activated on two robots without a localization capability, these robots must maintain the relative position to be able to execute simultaneously when sharing the same helping robot.

Meanwhile, to cope with dynamic and environmental influences for $R_j$, two questions must be answered: 1) *How should the robots adjust the coalition configuration $\boldsymbol{P}_j$ to maintain the affected sensor constraints?* 2) *What if the robots cannot adapt to these influences or the sensors fail, so that certain sensor constraints cannot be maintained?* Sections 4.1.2 and 4.1.3 answer the first question, while Sections 4.1.4 to 4.1.6 answer the second.

### 4.1.2 Assessing the Utility of $\mathbf{P}_{jk}$

Since each sensor constraint $C_{jk}$ is only associated with a subset of robots in $R_j$ (i.e., $R_{jk}$), it is natural to employ a localized approach. For maintaining $C_{jk}$, robots in $R_{jk}$ use the related information instance $F_{jk}(Ref_{1:N_{jk}})$. To consider dynamic and environmental influences, a measure is needed to quantify the utility of the multi-robot configuration $\mathbf{P}_{jk}$ for satisfying $C_{jk}$, which can also be interpreted as a quality measure of $F_{jk}(Ref_{1:N_{jk}})$. This measure is referred to as the *information quality measure* (IQ measure). The $i$th robot's configuration in $\mathbf{P}_{jk}$ is referred to as $\mathbf{P}_{jk}^i$. Without loss of generality, it is assumed that $r_{jk}^i$ is used to instantiate $Ref_i$ in $F_{jk}(Ref_{1:N_{jk}})$, in which $r_{jk}^i$ is the $i$th robot in $R_{jk}$.

The IQ measure given the multi-robot configuration $\mathbf{P}_{jk}$ for $r_{jk}^{1:N}$ is denoted as $Qual(F(r_{jk}^{1:N}) \mid \mathbf{P}_{Env})$, or abbreviated as $Qual(F \mid \mathbf{P}_{Env})$, in which $\mathbf{P}_{Env}$ represents the current local dynamic and

environmental situation. This measure is computed in two parts: the sensor quality measure $i(F)$ and the weight $w(F \mid \mathbf{P}_{Env})$. $Qual(F \mid \mathbf{P}_{Env})$ is then computed as $i(F) \cdot w(F \mid \mathbf{P}_{Env})$.

**Sensor Model**

The sensor quality measure $i(F)$ is computed using a given sensor model for a given $\mathbf{P}_{jk}$ for $r_{jk}^{1:N}$. Two essential submodels are explicitly identified – the sensor quality model and the sensor uncertainty model.

**Sensor Quality Model** The sensor quality model $i(F)$ is given as a function $I$: $\mathcal{P} \rightarrow [0, 1]$, in which $\mathcal{P}$ is the configuration space for $R_{jk}$. This function computes a mapping from $\mathbf{P}_{jk}$ to scores within $[0, 1]$. The scores are assigned according to how favorable the configuration is for retrieving $F(r_{jk}^{1:N})$ based on non-noise sensor characteristics (e.g., sensing range). A value of 1 indicates the most favorable configuration. Note that the influence of $r_{jk}^{S}$ can be ignored, since the sensor is located on $r_{jk}^{S}$.

Sensor quality models can be defined for laser and camera sensors in a 2D Euclidean space (consistent with [Murphy, 1998]) for the relative position information $F_R(r_1, r_2)$, given by:

$$I([l,\, \theta]_{r_1}) = a \cdot \frac{l_{max} - l}{l_{max}} + (1.0 - a) \cdot \frac{\theta_{max} - |\theta|}{\theta_{max}} \tag{4.1}$$

in which $[l,\, \theta]_{r_1}$ is the relative position vector from $r_1$ to $r_2$ (assuming the sensor is located on $r_2$) in a polar coordinate system and $l_{max}$, $\theta_{max}$ are the sensor distance and angle ranges, respectively. Here, $a$ is a weighting factor. Informally, this model prefers locations that are closer to the sensor, and to the midline of sight of the sensor. The sensor quality model for wireless positioning systems can be created similarly, although the angle parameter ($\theta$) may no longer be necessary.

**Sensor Uncertainty Model** This model captures the noise characteristics of the sensor, defined as a density function of sensor readings, $U$: $(\mathcal{P} \mid \mathcal{P}) \rightarrow \Re$, given the actual $\mathbf{P}_{jk}$. The model is defined as a linear normal distribution for laser and camera sensors (consistent with [Murphy, 1998] when

assuming Gaussian noise), given by:

$$U([l', \theta']_{r_1} \mid [l, \theta]_{r_1}) \sim N([l, \theta]_{r_1}, M\Sigma M^T) \tag{4.2}$$

in which $M$ is the scaling matrix and $\Sigma$ is the covariance matrix (for noise characteristics) for laser or camera,

$$\begin{matrix} |l - l'| & 0 \\ 0 & |\theta - \theta'| \end{matrix}$$

By separating these two models, it is implicitly assumed that the sensor quality model is conditionally independent of the sensor noise given the sensor uncertainty model. Relaxing this assumption is left for future work.

**Environment & Uncertainty Sampling**

The weight $w(F \mid \mathbf{P}_{Env})$ of the sensor quality measure reflects how dynamic factors and environment settings influence the information quality. To achieve this, a sampling method is used. First, different objects in the environment are represented by considering them as composed of samples. Next, the influence of these samples on the IQ measure are independently computed and then the influences are combined. One advantage of using a sampling method is that geometric reasoning is implicitly incorporated based on the approximated geometric representation created by the samples. By using this approach, it is assumed that each sample exerts influence independently. This assumption is generally true unless geometric structures have to be specifically modeled (e.g., finding triangle-shaped objects).

For sampling the environment, a $k$-means clustering algorithm is applied to range sensor readings, such that readings corresponding to different objects are more likely to fall into different clusters. Based on the environment's complexity with respect to the robots, a granularity (i.e.,

density of particles) is then chosen for sample creation. Finally, IQ-ASyMTR$^+$ samples again on these environment samples based on the sensor uncertainty model using the Metropolis-Hastings algorithm [Chib and Greenberg, 1995]. For each range sensor scan, environment sampling yields a set of $N$ samples, $S : \{s_1, s_2, ..., s_N\}$. For each environment sample $s_z$, after uncertainty sampling, a new set of samples is created, $S_z : \{s_z^1, s_z^2, ..., s_z^M\}$, for constant $M$. Figure 4.1(a) shows a scenario with $M = 3$.

**Computing the Weight**

Since environmental samples from $r_{jk}^{1:N}$ usually have no impact on the information quality (since these samples are part of the information itself), they are considered separately. For each environment sample $s_z$, the likelihood of the sample being $r_{jk}^i$ is computed, for all $i = 1..N$, as $\eta_z^i = \frac{U(s_z \,|\, \mathbf{P}_{jk}^i)}{Z}$, in which $s_z$ is also used to represent the position of $s_z$ in $r_{jk}^S$'s frame. $Z$ is a normalization constant. Then, the likelihood of $s_z$ being one of the robots is computed as $\eta_z = 1.0 - \prod_i (1.0 - \eta_z^i)$. For retrieving $F_R(r_1, r_2)$, this likelihood is $\eta_z = \eta_z^1 = \frac{U(s_z \,|\, \mathbf{p}_{jk}^1)}{Z}$ ($\eta_z^2 = 0$ when $r_{jk}^S = r_2$). When assuming that samples within a short distance (e.g., 0.2m) from $r_{jk}^i$ are samples from $r_{jk}^i$, $Z$ can be specified as $Z = U(\mathbf{P}_{jk}^i \pm c \,|\, \mathbf{P}_{jk}^i)$, where $c = [0.2, 0.2]^T$ when $\mathbf{P}_{jk}^i \in \Re^2$.

For $s_z^m \in S_z$, compute $h_z^m = H(s_z^m \,|\, \mathbf{P}_{jk})$, where $H$ is an application-specific function that captures the probability of risk from the uncertainty samples given $\mathbf{P}_{jk}$. The probability of risk for the associated environment sample $s_z$ is then combined as $\tau_z = C(h_z^1, h_z^2, ... h_z^M)$, where $C$ can also be implemented in an application-specific manner. A scenario for retrieving $F_R(r_1, r_2)$ is shown in Figure 4.1(b) where the probability of risk is computed as the ratio of the uncertainty samples falling in the risk range. In such a way, when there is an obstacle that is close to $r_1$, the samples of the obstacle increase the probability of risk towards the obstacle's direction.

The probability of risk for each sample $s_z$ is weighted by $1.0 - \eta_z$. The weight for $i(F)$ is computed as the joint probability of no risk considering all environment samples. Since independent samples are assumed, it holds that $w(F \,|\, \mathbf{P}_{Env}) = \prod_z (1.0 - \tau_z \cdot (1.0 - \eta_z))$. This measure maintains in $[0, 1]$.

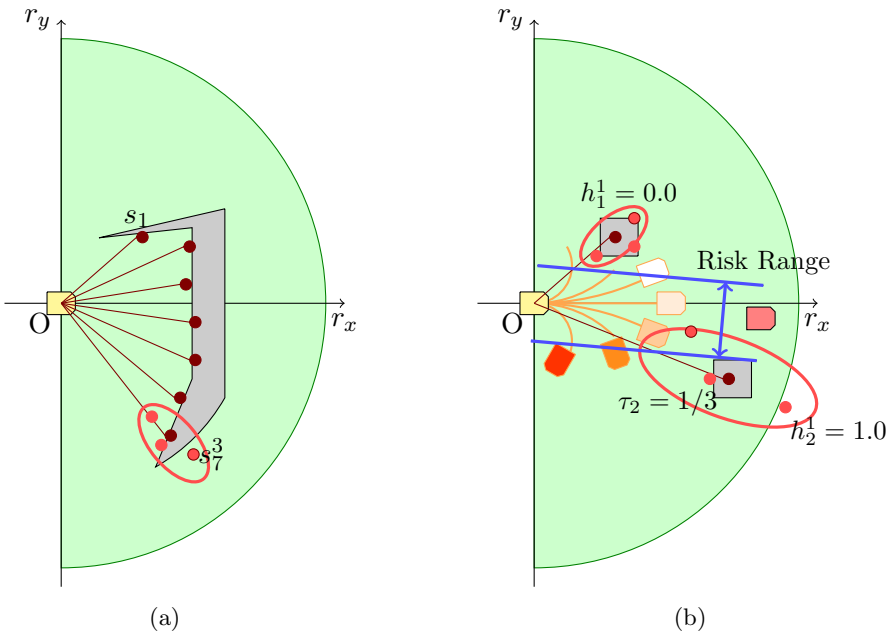



Figure 4.1: (a) Environment sampling using laser sensor with uncertainty sampling. Dark red (dark gray) particles are environment samples and bright red (light gray) particles at the bottom of the figure are uncertainty samples for one of the environment samples. (b) The probability of risk considering each environment sample is determined by the proportion of uncertainty samples falling in the risk range. The IQ measures are shown graphically for several velocity vectors. The higher the resulting information quality, the whiter the resulting configuration is drawn. Notice that although the sensor quality model would prefer the velocity vector leading straightly to the target, combined with the weight from environmental influence, the best choice is to curve a little to reduce the blocking risk from the nearest uncertainty sample.

### 4.1.3 Maintaining Sensor Constraints with the IQ Measure

During execution, the IQ measures for the sensor constraints are monitored. When this measure drops below a threshold (denoted by $\rho_1$) for any sensor constraint $C_{jk}$, a process described as follows is triggered on $r^S_{jk}$ to increase the measure. Given the localized nature of this approach, note that this approach cannot be proven stable, and hence is subject to local minima[2]. However, the sensor constraint relaxation process in Sections 4.1.4 to 4.1.6 can bump the execution out of a local minimum by establishing alternative sensor constraints.

#### Motion Model & Motion Sampling

The motion model is used to predict the resulting robot position given the current position and a command vector. This model can be specified as a function, $F_m : (Pos, V) \rightarrow Pos$, in which $Pos$ is the position space (e.g., $\Re^3$) and $V$ is the space of command vectors for the given motion model. In our implementations, the common differential drive motion model in a 2D space is used, which has the form $r = v/\omega$, where $r$ is the radius of movement, $v$ is the velocity and $\omega$ is the angular velocity. Both velocities are a linear function of the command vector.

$V$ is sampled into command vectors $\{v_1, v_2, ... v_D\}$ (consistent with [Simmons, 1996]), compute the corresponding velocity vectors, and choose $v^*$ with the best predicted IQ measure. Figure 4.1(b) explains this process in a simple scenario. This process is executed by $r^S_{jk}$ when the IQ measure drops below $\rho_1$.

#### The Algorithm

The algorithm for computing the desired command vector $v^*$ for improving the IQ measure for $F(r^{1:N}_{jk})$ is given in Algorithm 4. The algorithm starts with environment sampling. Then, for each command vector, the algorithm predicts the resulting IQ measure. The algorithm returns the command vector resulting in the best IQ measure in the current situation. The computation

[2]For example, robots in a coalition involving different sensor constraints may simultaneously choose to improve the IQ measure in different manners, which may negatively influence other robots during execution.

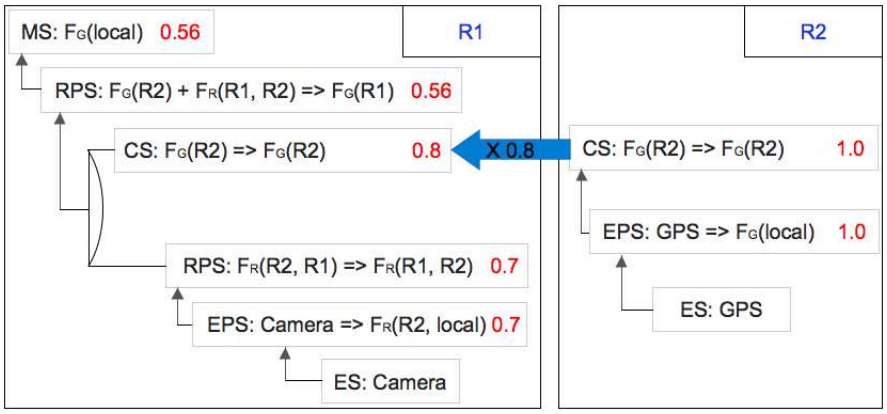



Figure 4.2: An instantiated potential solution from Figure 3.3 with the IQ measure.

of the algorithm is dominated by the nested *for* loops for computing the IQ measure, yielding a computation complexity of $O(M|V||S|)$.

### Incorporation of the IQ Measure

Figure 4.2 provides an example of the incorporation of the IQ measure. By assuming independence between different information instances, one can compute the IQ measure for information instances that are converted using RPSs as follows (in which $Qual(F_c \,|\, \mathbf{P}_{Env})$ is abbreviated as $Qual(F_c)$):

**Definition 4.1.1.** *Information Quality for Converted Information – For any RPS,* $F_1 + ... + F_N \Rightarrow F_c$*, it is defined that* $Qual(F_c) = \Pi_{i=1}^{N} Qual(F_i)$*.* □

This definition follows the intuition that the more dependencies there are, the less reliability there is. One can also scale down (using a fixed scalar or dynamically determined parameter) the measure when information is transferred, to account for the unreliability of communication. Note that when the same information is used multiple times in RPSs (either directly or indirectly), the measure of this information should be included in the computation only once.

**Algorithm 4** Compute $v^*$ to improve the IQ measure

Sample the environment, $S : \{s_1, s_2, ...\}$.
Sample $V$ into command vectors $\{v_1, v_2, ... v_D\}$.
Retrieve the current $\mathbf{P}_{jk}$ for $F(r_{jk}^{1:N})$.
**for all** $s_z$ in $S$ **do**
  Compute the likelihood $\eta_z = 1.0 - \prod_i (1.0 - \eta_z^i)$.
  Sample using the sensor uncertainty model of the range sensor, $S_z : \{s_z^1, ... s_z^M\}$.
**end for**
**for all** $v_q$ in $V$ **do**
  Predict $(\mathbf{P}_{jk})^q$ after executing $v_q$ given $\mathbf{P}_{jk}$ using $F_m$.
  Compute $i(F)^q = I((\mathbf{P}_{jk})^q)$.
  **for all** $s_z$ in $S$ **do**
    **for all** $s_z^m$ in $S_z$ **do**
      Compute $(h_z^m)^q = H(s_z^m \,|\, (\mathbf{P}_{jk})^q)$.
    **end for**
    Compute $\tau_z^q = C((h_z^1)^q, (h_i^2)^q, ...(h_i^M)^q)$.
  **end for**
  Compute $w(F \,|\, \mathbf{P}_{Env})^q = \prod_z (1.0 - \tau_z^q \cdot (1.0 - \eta_z))$.
  Compute $Qual(F \,|\, \mathbf{P}_{Env})^q = i(F)^q \cdot w(F \,|\, \mathbf{P}_{Env})^q$.
**end for**
Compute $Qual(F \,|\, \mathbf{P}_{Env})^* = \max_{i=1}^{D} (Qual(F \,|\, \mathbf{P}_{Env})^q)$.
**return** $v^*$ that leads to $Qual(F \,|\, \mathbf{P}_{Env})^*$.

### 4.1.4 Expressivity of Information for Constraint Relaxation

In this and the following two subsections, the second question from Section 4.1.1 is answered: *What if the robots cannot adapt to dynamic and environmental influences or the sensors fail, so that certain sensor constraints cannot be maintained?* In IQ-ASyMTRe$^+$, these situations can be identified in a uniform manner by checking whether the IQ measure for any sensor constraint drops below a threshold $\rho_2$, for $\rho_2 \in [0, \rho_1)$. In such cases, it is investigated that how the unsatisfied constraints can be relaxed by using redundancies at sensory and computational levels to achieve more robustness during execution. The motivation is that the required information can often be retrieved in different ways by the robots in the original coalitions assigned to the task, since it may be converted from other available information. While a more intuitive and simpler approach is to re-allocate the task to new coalitions, such a process can be computationally expensive. Furthermore, it is often difficult to bring new robot coalitions together, since robots in these coalitions may be physically distant from each other. Moreover, there may not always be other robots available. In IQ-ASyMTRe$^+$, constraint relaxation is achieved by using IQ-ASyMTRe [Zhang and Parker, 2010b] locally to search for alternative interactions using robots in the current coalitions for the task. Instead of running IQ-ASyMTRe on all available robots in the system to reallocate the task, IQ-ASyMTRe$^+$ works only with robots in the current coalitions. This results in more efficiency for processes that run exponentially in the number of robots.

As with forming initial formations, constraint relaxation is applied to the inputs of the required MSs. However, note that the capability of such a process can be limited by how the MSs are implemented. In different applications, MSs that achieve the same behavior are often implemented to require different inputs. These implementations take advantage of prior knowledge about the task or the capabilities of the robots, while suffering generality. These MSs demonstrate the same motor behavior, which is referred as their *MS class*. A process is introduced that can autonomously exploit the available MSs of the same class for more generality, which can be proven to provide more flexibility and robustness. The process is based on the idea that all MSs in the same MS

class should be associated with one common notation – the minimum information requirement. The notion of *information requirement* is first introduced, which specifies the required input information for any process. In IQ-ASyMTRe$^+$, the information requirement for any MS is specified as an IIS defined as follows:

**Definition 4.1.2.** *Information Instance Set (IIS) – An IIS is a set of information instances, and can be represented as* $\{F_1, F_2, F_3, ...\}$. *All information instances within the set are connected with the AND condition.* □

Assuming that information instances have unique semantic meanings[3], it can be shown that the semantic meaning of any information requirement can be expressed using IISs. Let us first define the notion of *information configuration*. An information configuration differs from a physical robot configuration that is used previously; it refers to a configuration that is specified solely by the semantic meaning of the given information. For example, $F_G(r_1)$ refers to the global position of $r_1$ but it does not specify where exactly $r_1$ is at.

**Lemma 4.1.2.** *The semantic meaning (the information configuration) expressed by any information requirement can be expressed using IISs connected with OR.*

*Proof.* Note that by definition, an information instance is a complete reference of information. Hence, fully instantiated information instances express the most specific information configurations of the related entities in the environment, given the above assumption. Consider any IIS as describing the joint information configuration of the entities in the environment, and consider IISs connected with *OR* as an union of these joint information configurations.

Furthermore, any information requirement can be considered as a black box that checks the joint information configuration of entities and outputs whether the configuration satisfies the requirement. Given the related information types and number of information instances for each type

[3]The uniqueness requires that the semantic meaning expressed by one information instance cannot be fully expressed by another information instance, unless they are semantically equivalent (i.e., convertible from each other).

necessary for expressing the information requirement (i.e., from domain knowledge), the information requirement can be constructed as follows.

For every possible way to fully instantiate the related information instances, an IIS can be created. Input this IIS to the black box which returns whether it satisfies the requirement. Since these joint information configurations represent the most specific configurations given the related information instances (each information instance imposes an additional specification on the joint information configuration, since they are connected with *AND*), and given that every information instance is also the most specific based on the previous conclusion, all possible joint information configurations that satisfy the information requirement would be checked. Finally, the IISs that satisfy the information requirement only need to be connected with *OR* to create the representation. □

For example, for an information requirement that requires the global position of the local robot or the relative position between the local robot and robot $R_1$, from domain knowledge, it holds that the related information instances are $F_G$ and $F_R$. For simplicity, it is also assumed that there are only three entities in the environment – the local robot, $R_1$, and $R_2$. Based on the process in the proof, the output is $\{F_G(local),\ F_R(R_1, R_2)\}$ *OR* $\{F_G(local),\ F_R(R_2, R_1)\}$ *OR* $\{F_G(local),\ F_R(R_1, local)\}$ *OR* $\{F_G(local),\ F_R(local, R_1)\}$ *OR* $\{F_G(local),\ F_R(R_2, local)\}$ *OR* $\{F_G(local),\ F_R(local, R_2)\}$ *OR* $\{F_G(R_1),\ F_R(R_1, local)\}$ *OR* $\{F_G(R_1),\ F_R(local, R_1)\}$ *OR* $\{F_G(R_2),\ F_R(R_1, local)\}$ *OR* $\{F_G(R_2),\ F_R(local, R_1)\}$, which is equivalent to the information requirement, given that there are only three entities in the environment. Furthermore, the following property holds regarding the use of IIS to express information requirements.

**Lemma 4.1.3.** *Given that the entity set can be countably infinite, if an information requirement can be specified with a finite representation that uses IISs connected with OR, the representation is unique.*

*Proof.* The proof is a constructive one. Once the representation for the information requirement is obtained using the previous process, the IISs using uninstantiated referents can simply be combined

(assuming that one can check whether the entire entity set is covered or not). The newly created IISs join the combination process immediately. When there are no more distinct combinations possible, the process terminates and any IISs that have been used at least once in the combinations can be simply removed. Note that the same IIS can be used multiple times. Finally, each IIS can be simplified 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 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 input is unique, the final representation must also be unique. □

In the example above, the first six IISs can be combined as $\{F_G(local),\ F_R(X, Y)\}$; $\{F_G(local),\ F_R(R_1, local)\}$, $\{F_G(local),\ F_R(local, R_1)\}$, and the last four IISs can be combined as $\{F_G(X),\ F_R(local, R_1)\}$ *OR* $\{F_G(X),\ F_R(R_1, local)\}$. After removing information instances that have no instantiated referents and no referent instantiation constraints, the final representation becomes $\{F_G(local)\}$ *OR* $\{F_R(local, R_1)\}$ *OR* $\{F_R(R_1, local)\}$. This representation is much more concise compared to the input.

### 4.1.5 Minimum Information Requirement

Generally, the more information a MS requires, the more difficult it is to activate it. Nevertheless, different MSs are often implemented to work with specific systems, which results in extra specifications imposed on the information requirements. For example, to satisfy the MS class of navigating to a relative position (referred to as *goto-relative*), one MS for a system in which robots can localize may conveniently use $\{F_G(local), F_G(goal)\}$, while a MS working with an overhead camera system may require $\{F_R(X, local), F_R(X, goal)\}$ (i.e., $X$ can be any camera).

To achieve more generality, it is necessary to identify and remove extra specifications so that the information requirements become minimally sufficient for the MS classes. First, a definition of *minimum information requirement* is provided.

**Definition 4.1.3.** *Minimum Information Requirement – A minimum information requirement is an information requirement for a MS class that satisfies:*

- *Removing any specification from the information configuration specified by the requirement makes it insufficient.*
- *Imposing more specification on the information configuration specified by the requirement does not influence its sufficiency.* □

For example, suppose that a minimum information requirement can be specified as $\mathcal{E} = \{F_R(R_2, R_1), F_G(R_1)\}$. While removing $F_R(R_2, R_1)$ would make $\mathcal{E}$ insufficient, adding $F_G(R_2)$ to it does not influence its sufficiency. However, note that the definition does not imply the sufficiency of $\mathcal{E}$, after simultaneously removing $F_R(R_2, R_1)$ and adding $F_G(R_2)$. Furthermore, one should also require that any forms that represent a minimum information requirement of a MS class are semantically equivalent. This requirement ensures that the minimum information requirement for any MS class is unambiguously defined, such that the systems that are capable of satisfying the MS class are equivalent information systems [Donald et al., 1997] for the class. As an example, using IISs, suppose that $\{F_G(X)\}$ is the information requirement for a MS class. Although $\{F_G(R_1)\}$ satisfies the requirements in the definition, it is not a minimum information requirement, as $\{F_G(R_2)\}$ is not semantically equivalent to $\{F_G(R_1)\}$.

In the following, the effects of this definition is investigated for using IISs in IQ-ASyMTRe$^+$. First, semantic equivalence needs to be defined for IISs. Based on the definition, if one can infer the information configuration of entities specified by some information from other information and vice versa, they are semantically equivalent. In our approach, the inference between information is performed using information conversions. Hence, semantic equivalence is defined accordingly.

**Definition 4.1.4.** *Reduction of IIS – For any two IISs, $s_1$ is 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 4.1.5.** *Semantic Equivalence of IIS – Two IISs ($s_1$ and $s_2$) are semantically 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 semantically equivalent to $\{F_G(r_A), F_G(r_B)\}$ given this definition. The $\succ$ operator defines a partial ordering for IISs. The following definitions are further introduced.

**Definition 4.1.6.** *Power Set of IIS – The power set of any IIS $s$, (denoted by $P(s)$), includes $s$ and all information instances that can be converted from $s$. An IIS that cannot produce new information instances using information conversions is called a maximum IIS (MaxIIS).* □

For example, for one MS of the *goto-relative* class, it holds that $P(\{F_G(local),\ F_G(goal)\})$ $= \{F_G(local),\ F_G(goal),\ F_R(local, goal),\ F_R(goal, local)\}$, and $P(\{F_R(X, local),\ F_R(X, goal)\}) = \{F_R(X, local),\ F_R(local, X),\ F_R(X, goal),\ F_R(goal, X),\ F_R(local, goal),\ F_R(goal, local)\}$. Notice that the power set of any IIS is also a MaxIIS accordingly.

**Definition 4.1.7.** *Kernel IIS – For any IIS $s$, the kernel IIS (denoted by $K(s)$), can be any subset of $P(s)$, such that: any information instance in $K(s)$ cannot be converted from any other information instances in $K(s)$, and $K(s) \succ s$.* □

One kernel IIS for $\{F_G(robot), F_G(goal), F_R(goal, robot), F_R(robot, goal)\}$ is $\{F_G(robot), F_G(goal)\}$.

**Definition 4.1.8.** *Minimum IIS – The minimum IIS (MinIIS) (denoted as $s_{min}$) for a MS class is a MaxIIS which satisfies: for any IIS $s$ that satisfies the information requirement of the MS class, it holds that $s_{min} \subseteq P(s)$ and $s_{min}$ satisfies it as well.* □

The following theorem shows the relationships between the minimum information requirement and the MinIIS.

**Theorem 4.1.4.** *For any MS class, the MinIIS exists; when it has a finite representation, the representation is unique.*

*Proof.* From Lemma 4.1.2, given a MS class, it holds that a representation in the form of $\{\cup_k \text{IIS}_k\}$ can be used to express the information requirement that includes all information configurations specified by any minimum information requirements and only these configurations, in which $\cup$ represents the *OR* condition. It is straight-forward to conclude that any minimum information requirement is included in an IIS in $\{\cup_k \text{IIS}_k\}$ as *OR* conditions are naturally absent. In other words, any IIS in $\{\cup_k \text{IIS}_k\}$ includes one or more minimum information requirements for the class. From Definition 4.1.3, one can infer that all IISs in $\{\cup_k \text{IIS}_k\}$ are semantically equivalent and hence their power sets are the same. This power set is the MinIIS for the class since any IIS that satisfies the information requirement must at least include one of the minimum information requirements. From Lemma 4.1.3, when this MinIIS has a finite representation, it is unique. □

Since MinIIS represents the information that is commonly available in all IISs for satisfying the MS class, it can be easily approximated using Algorithm 5. The computational complexity is given as follows. Among all IISs for the implemented MSs of the MS class, denote the maximum number of related information types as $N_t$, the maximum number of distinct entities as $N_e$, the maximum number of referents for the related information types as $N_r$, the maximum number of RPSs producing the same information type as $N_c$, the maximum number of information instances in any RPSs as $N_i$, and the number of the implemented MSs as $N_{MS}$. The number of distinct information instances for any IIS is bounded by $N_t N_e^{N_r}$; the complexity to check each information instance is bounded by $N_c N_e^{N_r N_i}$. Hence, the complexity of Algorithm 5 is $O(N_{MS} N_c N_t N_e^{N_r(N_i+1)})$.

**Algorithm 5** Approx. the $MinIIS$ using IISs for MSs

**for all** $IIS_i \in$ IISs for the implemented MSs **do**
  Compute $S_i = P(IIS_i)$.
**end for**
**return** $S = \cap_i(S_i)$.

For the *goto-relative* MS class, the algorithm would output $\{F_R(goal, local), F_R(local, goal)\}$. When applying constraint relaxation on the MinIIS for the MS class, more alternative ways to satisfy the information requirement can be found:

**Corollary 4.1.5.** *Expressing the information requirement using any kernel set of the MinIIS for a MS class maximizes the number of distinct potential solutions.*

*Proof.* First, it is easy to see that any kernel IISs of the MinIIS are semantically equivalent. Hence, any IIS that satisfies one would also satisfy the others. Furthermore, for any IIS $s$ satisfying the information requirement, according to Definition 4.1.8, it holds that $\tilde{s} \subseteq P(s)$ ($\tilde{s}$ is any kernel IIS of the MinIIS). Hence, any IIS that satisfies $s$ would also satisfy $\tilde{s}$. Thus, the number of potential solutions with $\tilde{s}$ cannot be less than with any IIS for the MS class. □

### 4.1.6 Constraint Relaxation

More flexible and robust execution can be achieved by performing constraint relaxation on any kernel set of the approximated MinIIS for the MS class. The tradeoff, however, is that the time to process the potential solutions also increases. While the size of the solution space for reasoning about an IIS with a single information instance is restricted to be exponential in some constants given the problem domain, the size is exponential in the size of the IISs (i.e., the number of information instances) [Zhang and Parker, 2012d]. The reason is due to the constraint that requires referents with the same labels to be instantiated to the same entities. For example, when the input is $\{F_G(X), F_R(X, local)\}$, any solution for retrieving $F_G(X)$ must be checked with any solution for retrieving $F_R(X, local)$.

An intuitive method to address this issue is inspired by the idea that independencies among random variables are used to restrict the exponential growth of the joint probability tables. Similar independence relationships between information instances are identified in the following definition.

**Definition 4.1.9.** *Independence of Information Instance – An information instance is independent of another if there are no uninstantiated referents labeled the same.* □

The advantage for two information instances that are independent is that they can be reasoned about separately, since the instantiation of either one would not conflict with the other. The notion of independence of information instances can be easily extended to IISs and can be used to divide any IIS into mutually independent IISs. For example, $\{F_R(R_1, X), F_R(R_2, X), F_G(local)\}$ can be divided into $\{F_R(R_1, X), F_R(R_2, X)\}$ and $\{F_G(local)\}$. In such a way, the size of the solution space can be significantly reduced. Suppose that all information instances in the original IIS are mutually independent so that every one becomes an IIS. Then, the size of the solution space can be reduced from $\Pi_{k\in[1:K]}N_k$ to $\sum_{k\in[1:K]}N_k$, in which $N_k$ represents the size of the solution space for each information instance.

Given an IIS to reason about, it can first be divided into multiple mutually independent IISs when applicable. Note that for the IIS of $\{F_R(R_1, local), F_R(R_2, local)\}$, the independence trivially holds since there are no uninstantiated referents. In most applications, since the required information instances are almost always fully instantiated, the solution spaces can be reduced from exponential to linear.

## 4.2 IQ-ASyMTRe$^+$: The Algorithms

The IQ-ASyMTRe$^+$ architecture uses IQ-ASyMTRe to model and identify the interactions between the robots and incorporates the approaches discussed in this research to achieve flexible and robust execution for a tightly-coupled multi-robot task. Algorithm 6 reasons about the required interactions; this algorithm is used both for initially forming the coalitions and for relaxing sensor constraints during execution. To distinguish the differences from IQ-ASyMTRe, the parts of the algorithm introduced by IQ-ASyMTRe$^+$ is italicized. This algorithm is quadratic in the size of the solution space for a full search of the space.

Algorithm 7 is the main algorithm. At the very beginning, every robot that is assigned a MS class to execute invokes the IQ-ASyMTRe$^+$ reasoning algorithm to form an initial coalition. During the execution, the IQ measure of the required information for these coalitions is continuously

**Algorithm 6** IQ-ASyMTRe$^+$: The Reasoning Algorithm

(**Text in italics are major differences compared to [Zhang and Parker, 2010b].**)
*Approximate MinIIS from the available MSs using Alg. 5.*
*Choose a kernel set of the approximated MinIIS as the input.*
*Divide the inputs into mutually independent sets.*
*For each set*, reason about the required information to create the solution spaces using IQ-ASyMTRe.
*For each set*, extract and order potential solutions (PoSs) in a list based on predefined costs for schemas.
**while** *not all independent sets are satisfied* **or** the given search time has not elapsed **do**
  **for all** *unsatisfied independent sets* **do**
    **if** not all PoSs are checked **then**
      Retrieve the next PoS on the ordered list.
      Activate the required ES-EPS nodes temporarily in the PoS to collect sensory information.
      Activate CS nodes in the PoS to send information requests for the information instances.
    **end if**
    Collect information communicated by other robots.
    Process requests and share information.
    **for all** checked PoSs **do**
      **if** the required information is available **then**
        Compute the cost and record the coalition.
      **end if**
    **end for**
  **end for**
**end while**
*For each set*, set up the coalition with the least cost, and *with information of the best IQ measure.*
**return** The coalition.

monitored. Whenever the measure is good enough (i.e., above $\rho_1$), the robots execute the command vector of the MS class; otherwise, they compute and execute $v^*$ using Algorithm 4 (when the information is retrieved using local sensors), or request robots with the sensors to improve the IQ measure. Whenever the robots have difficulties to adapt to dynamic and environmental influences using Algorithm 4 (i.e., the IQ measure drops below $\rho_2$) or the information becomes unavailable due to sensor failures (i.e., the IQ measure becomes 0), the reasoning process is re-triggered locally for constraint relaxation.

**Algorithm 7** IQ-ASyMTRe$^+$: The Main Algorithm

```
Invoke Alg. 6 to form a coalition for the MS class.
Invoke Alg. 3 to coordinate robots to maintain constraints.
while true do
  Obtain IQ measure for the required sensor constraints.
  if an IQ measure is below ρ2 then
    Invoke Alg. 6 to search alternative solutions.
  else if an IQ measure is below ρ1 then
    Request or invoke Alg. 4 to improve the IQ measure.
  else
    Execute the command vector of the MS class.
  end if
end while
```

# Chapter 5

# Task allocation

When there are multiple tasks to be assigned, the assignment of one task may influence other tasks. Task allocation addresses the problem of how to assign tasks to robots to achieve better overall system performance. This chapter for task allocation contains the following discussions [Zhang and Parker, 2012a]. After presenting a general formulation of the ST-MR-IA problem in Section 5.1, a formal analysis of two natural greedy heuristics for ST-MR-IA is provided in Section 5.2. A new greedy heuristic and the algorithms for implementing it are discussed in Section 5.3. The extended formulation of the ST-MR-IA problem (referred to as ST-MR-IA-TD) and the result on the hardness of approximating it are presented in Section 5.4. An algorithm that utilizes the discussed methods for ST-MR-IA to address the ST-MR-IA-TD problem is provided in the same section. Simulation results for these two formulations of the problem are presented in Sections 7.3 and 7.4, respectively.

## 5.1 Problem formulation

First, a general formulation of the ST-MR-IA problem is provided. This problem is often constructed similarly [Service and Adams, 2011, Shehory and Kraus, 1998, Vig and Adams, 2006] as follows:

Given:

- a set of robots, $R = \{r_1, r_2, ...\}$. Each robot $r_i$ is associated with a vector $\boldsymbol{B}_i$ of $H$ real non-negative capabilities, in which $H$ is assumed to be a constant.
- a set of coalitions, $C = \{c_1, c_2, ...\}$. Each coalition $c_j$ satisfies $c_j \subseteq R$.
- a set of tasks to be assigned, $T = \{t_1, t_2, ...\}$. Each task $t_l$ requires a vector $\boldsymbol{P}_l$ of $H$ real non-negative capabilities.
- a vector $\boldsymbol{W}$ of real non-negative costs for capabilities: the use of the capability indexed by $h$ incurs $\boldsymbol{W}[h]$ cost per unit.
- a vector $\boldsymbol{V}$ of real positive rewards for tasks: accomplishing task $t_l$ receives $\boldsymbol{V}[l]$ reward.
- a function $Cost : C \times T \rightarrow \Re^0$ that computes real non-negative communication and coordination costs for assignments based on the coalition-task pair.
- a utility function $U$ for assignments, defined as:

$$U(m_{jl}) = \begin{cases} \boldsymbol{V}[l] - \sum_h \boldsymbol{P}_l[h]\,\boldsymbol{W}[h] - Cost(c_j, t_l) & \text{if } \forall h : \sum_{r_i \in c_j} \boldsymbol{B}_i[h] \geq \boldsymbol{P}_l[h], \\ 0 & \text{otherwise.} \end{cases}$$

in which $m_{jl}$ denotes the assignment of $c_j \rightarrow t_l$. Note that although mathematically, $\boldsymbol{V}[l]$ and $\sum_h \boldsymbol{P}_l[h]\,\boldsymbol{W}[h]$ can be combined for each task $t_l$ as a single measure, they are often independent in robotic applications, and hence are specifically modeled for more generality[1].

Then the problem is to maximize:

$$\sum_j \sum_l U(m_{jl})\beta_{jl} \tag{5.1}$$

[1]Although not investigated in this research, the costs of the capabilities may be dependent on the robots. For example, if the cost is related to the time spent to perform a computation, a robot with a faster processor should incur a lower cost.

subject to the constraints:

$$\sum_{j}\sum_{l}\alpha_{ij}\beta_{jl} \leq 1 \quad \forall r_i \in R$$
$$\sum_{j}\beta_{jl} \leq 1 \quad \forall t_l \in T \tag{5.2}$$

in which $\beta_{jl}$ is 1 if $m_{jl}$ is in the chosen assignments or 0 otherwise, and $\alpha_{ij}$ is 1 if $r_i \in c_j$ or 0 otherwise. Note that the first constraint also implies that a coalition can be assigned to no more than one task in the chosen assignments.

Any assignment $m_{jl}$ that satisfies $\forall h : \sum_{r_i \in c_j} \boldsymbol{B}_i[h] \geq \boldsymbol{P}_l[h]$ is referred to as a *feasible* assignment. In this research, it is assumed that the utility function $U$ always returns positive values for feasible assignments (to distinguish from infeasible assignments). Note that one can simply ignore feasible assignments for which the overall costs are no less than the rewards of the tasks, and for which $U(m_{jl})$ is non-positive, since they would not increase the solution quality. Henceforth, when referring to assignments, it always refers to feasible assignments. For example, when stating that no assignments exist, it really means that no feasible assignments exist. Another note is that while $|C|$ can be exponential in the number robots (i.e., $2^{|R|}-1$), which also leads to an exponential space complexity for $Cost$, reasonable assumptions are often utilized (e.g., [Shehory and Kraus, 1998]) to restrict $|C|$.

## 5.2 Natural greedy heuristics

In this section, two natural greedy heuristics are presented for addressing the ST-MR-IA problem and an analysis of their performances is provided. Before continuing, a formal definition of *worst case ratio* (similar to the definition of *approximation factor* in [Service and Adams, 2011] or *ratio bound* in [Shehory and Kraus, 1998]) is provided, which is used to describe the quality of approximations. In the definition, $\boldsymbol{f}$ is used to denote any computable function.

**Definition 5.2.1.** *Given a maximization problem with solutions having positive values, an approximation algorithm has a worst case ratio $\theta = \boldsymbol{f}(I)$ $(\theta \geq 1)$, if it satisfies $S^*(I) \leq \theta \cdot S(I)$ for any problem instance of $I$, in which $S^*(I)$ is the value of the optimal solution and $S(I)$ is the value of the solution produced by the algorithm. When $\boldsymbol{f}$ is a polynomial time computable function, the worst case ratio is also referred to as a poly-time worst case ratio.* □

### 5.2.1 *AverageUtility*

The *AverageUtility* heuristic at each step chooses the assignment that maximizes the average utility per robot, until no more assignments that satisfy the constraints in Equation 5.2 exist. More formally, at step $\lambda$, denote the previously chosen set of assignments as $G_{\lambda-1}$. *AverageUtility* chooses the assignment $m_{pq}$ that satisfies the problem constraints (given that $\forall m_{jl} \in G_{\lambda-1} : \beta_{jl} = 1$) while maximizing $\frac{U(m_{pq})}{|c_p|}$. The following theorem establishes the worst case ratios of this heuristic.

**Theorem 5.2.1.** *Applying AverageUtility to the ST-MR-IA problem yields a worst case ratio $\theta = |R|$ without restricting the size of the coalitions. Furthermore, restricting the maximum size of the coalitions to be $k$ gives a worst case ratio $\theta = 2k$.*

*Proof.* At the beginning of any greedy step $\lambda$, denote the remaining set of robots as $R_\lambda$, the remaining set of tasks as $T_\lambda$, and the assignment to be chosen by *AverageUtility* as $m^\lambda = (c^\lambda \rightarrow t^\lambda)$. According to the greedy criterion, $m^\lambda$ has the maximum utility per robot in the remaining problem of $(R_\lambda, T_\lambda)$. As a result, the optimal solution for $(R_\lambda, T_\lambda)$ yields an overall utility of no more than $|R_\lambda|\frac{U(m^\lambda)}{|c^\lambda|}$. This upper bound is reached when all assignments in the optimal solution for solving $(R_\lambda, T_\lambda)$ have a utility per robot of no less than[2] $\frac{U(m^\lambda)}{|c^\lambda|}$ and every robot in $R_\lambda$ is assigned to a task. Hence, the worst case ratio for solving $(R_\lambda, T_\lambda)$ is $\frac{|R_\lambda|}{|c^\lambda|}$. As this is true for every step, it holds true in particular for $(R_1, C_1) = (R, T)$. Consequently, the worst case ratio for *AverageUtility* can be no worse than $|R|$, given that $|c^1| \geq 1$.

[2]They cannot have more due to the greedy criterion.

When the maximum size of the coalitions is restricted to be $k$, an induction process can be applied on the sizes of the robot and task sets. Suppose that the worst case ratio $2k$ holds for solving $(R', T')$ in which $R' \subseteq R$, $T' \subseteq T$ and the equalities do not hold simultaneously. For solving $(R, T)$, denote the first assignment made by *AverageUtility* as $m^1$, which has the maximum utility per robot. Denote the set of overlapping assignments[3] with $m^1$ in the optimal solution for solving $(R, T)$ as $M^*$ and the set of tasks in $M^*$ as $T^*$. As each robot can be assigned to at most one task, the following holds:

$$|M^*| \leq |c^1| \tag{5.3}$$

Use $R^*$ to denote all robots in $M^*$, and $R^+$ to denote all robots in $M^*$ or $c^1$. (Note that a robot in $c^1$ may not be in $M^*$, since the robot may not be used in the optimal solution.) Given the monotonicity of the optimal solution[4], the following must hold:

$$S^*(R - R^+, T - T^*) \leq S^*(R - c^1, T) \tag{5.4}$$

recall that $S^*(I)$ represents the optimal solution for $I$.

From the assumption of the induction, it holds that:

$$S^*(R - c^1, T - t^1) \leq 2k \cdot S^{AU}(R - c^1, T - t^1) \tag{5.5}$$

in which $S^{AU}(I)$ is used to denote the solution returned by *AverageUtility* for $I$. Also note that $(R - R^*, T - T^*)$ is a subproblem for $(R, T)$ (so is $(R^*, T^*)$), in that if the set of assignments using $R - R^*$ and involving $T - T^*$ in the optimal solution for $(R, T)$ yields a lesser or equal overall utility, it can be directly substituted by the set of assignments in the optimal solution for $(R - R^*, T - T^*)$

[3]For two assignments $m_{jl}$ and $m_{pq}$, $m_{jl}$ overlaps with $m_{pq}$ (or vice versa) if $c_j \cap c_p \neq \emptyset$.

[4]Given $(R_1, T_1)$ and $(R_2, T_2)$, if $R_1 \subseteq R_2$ and $T_1 \subseteq T_2$, it must hold that the overall utility of the optimal solution for $(R_2, T_2)$ is no less than for $(R_1, T_1)$. In fact, choosing the same set of assignments for $(R_2, T_2)$ as in the optimal solution for $(R_1, T_1)$ would yield the same overall utility for $(R_1, T_1)$ and $(R_2, T_2)$.

to create a better or equivalent solution[5]. Since robots in $R^+ - R^*$ are not present in the optimal solution for $(R, T)$, it must hold that $S^*(R - R^+, T - T^*) = S^*(R - R^*, T - T^*)$. Furthermore, for solving $(R^*, T^*)$, the optimal solution obtains a utility no more than $k \cdot |M^*| \cdot \frac{U(m^1)}{|c^1|}$, which happens only when every task in $T^*$ is assigned with utility per robot no less than $\frac{U(m^1)}{|c^1|}$ and is assigned to a coalition with exactly $k$ robots. Hence, the following holds:

$$S^*(R, T) \leq k \cdot |M^*| \cdot \frac{U(m^1)}{|c^1|} + S^*(R - R^+, T - T^*) \tag{5.6}$$

From the above equations, it can be concluded that:

$$\begin{aligned} S^*(R, T) &\leq k \cdot U(m^1) + S^*(R - c^1, T) \\ &\leq k \cdot U(m^1) + k \cdot \frac{U(m^1)}{|c^1|} + S^*(R - c^1, T - t^1) \\ &\leq 2k \cdot U(m^1) + 2k \cdot S^{AU}(R - c^1, T - t^1) \\ &\leq 2k \cdot S^{AU}(R, T) \end{aligned} \tag{5.7}$$

Finally, as the induction assumption holds trivially when $|R'| \leq 1$ and $|T'| \leq 1$, the conclusion holds. □

Note that when $k$ is relatively close to $|R|$, the worst case ratio for the restricted case is in fact better than $2k$. This is due to the fact that the inequalities in the proof can be further tightened in these situations. For example, when $k = |R|$ (equivalent to the unrestricted case), the worst case ratio for the restricted case is $|R|$ instead of $2|R|$. Similar effects can also be discerned in the analysis for the following heuristic.

[5]This substitution does not influence the assignments involving $T^*$ in the optimal solution for $(R, T)$, since all robots involving $T^*$ are in $R^*$.

### 5.2.2 *MaxUtility*

The *MaxUtility* heuristic at each step chooses the assignment with the maximum utility, until no more assignments that satisfy the constraints in Equation 5.2 exist. More formally, at step $\lambda$, denote the previously chosen set of assignments as $G_{\lambda-1}$. *MaxUtility* chooses the assignment $m_{pq}$ that satisfies the problem constraints (given that $\forall m_{jl} \in G_{\lambda-1} : \beta_{jl} = 1$) while maximizing $U(m_{pq})$.

**Theorem 5.2.2.** *Applying MaxUtility to the ST-MR-IA problem yields a worst case ratio $\theta = |R|$ without restricting the size of the coalitions. Furthermore, restricting the maximum size of the coalitions to be $k$ gives a worst case ratio of $\theta = k + 1$ [Service and Adams, 2011].*

*Proof.* Service and Adams [Service and Adams, 2011] have proven these worst case ratios for *MaxUtility*. It is also not difficult to conclude the same using a similar induction process as shown in the previous proof. □

First of all, it is important to note that algorithms for implementing both heuristics are exponential in the number of robots (i.e., $|R|$) when the maximum size of the coalitions is not restricted and are polynomial in the order of $O(|R|^k)$ when it is restricted to $k$. Furthermore, it may appear at first that *AverageUtility* should yield a better worst case ratio than *MaxUtility* in the restricted case, although the theoretical results turn out to be quite to the contrary. Another note is that the worst cases in the above proofs can actually occur, such that all these proven worst case ratios are in fact tight bounds. Although it is shown that approximation algorithms with a worst case ratio asymptotically no worse than $k/\log(k)$ (which is already close to the worst case ratios of the two natural heuristics) are unlikely to exist unless $P \equiv NP$, it does not imply that algorithms with better average performance, or with better worst case ratios for certain problem instances, cannot be found.

## 5.3 The new greedy heuristic

The inspiration to create a new heuristic with better average performance comes from the two natural heuristics. Although *MaxUtility* has a better worst case ratio than *AverageUtility* in the restricted case, as the result sections show, the two heuristics actually perform similarly empirically. Our explanation for this phenomenon can be understood in the proofs of their worst case ratios. To achieve the worst case ratio for *AverageUtility*, the problem instance has to be more constrained than for *MaxUtility*. In other words, it is less likely for a worst case scenario to occur for *AverageUtility* than for *MaxUtility*. Keeping this in mind, a new heuristic is presented that considers inter-task resource constraints to address the ST-MR-IA problem. Instead of making greedy choices based solely on the assignment, the new greedy heuristic also considers the influence between different assignments for task allocation.

### 5.3.1 A motivating example

As a motivating example, consider the case when there are four tasks $T = \{t_1, t_2, t_3, t_4\}$ with capability requirements $\boldsymbol{P}_1 = (1, 1, 1, 0, 0)$, $\boldsymbol{P}_2 = (1, 0, 0, 1, 1)$, $\boldsymbol{P}_3 = (0, 1, 0, 1, 1)$, $\boldsymbol{P}_4 = (0, 0, 1, 1, 1)$. Suppose that each robot has one and only one capability with a non-zero value (i.e., 1). Furthermore, suppose that there are sufficient robots for the last two capabilities and there is only one robot capable for each of the first three. Let the costs for the capabilities be the same and let $Cost$ return zeros for all assignments. In this scenario, when $t_1$ has a slightly higher reward than any other task, both *AverageUtility* and *MaxUtility* produce a solution in which only $t_1$ is assigned, hence giving a solution with poor quality. A better solution is to assign each of the three robots to tasks $t_2$, $t_3$ and $t_4$, respectively, which collectively yield a greater utility.

### 5.3.2 Inter-task resource constraints

From the previous example, it can be concluded that one problem with the natural heuristics is that a task with a slightly higher reward may be assigned to robots that are essential for other tasks,

which in turn sabotages the assignments of these other tasks. In light of this, following definition is provided.

**Definition 5.3.1.** *For any two assignments $m_{jl}$ and $m_{pq}$, $m_{jl}$ conflicts with $m_{pq}$ (or vice versa) if $c_j \cap c_p \neq \emptyset$ or $t_l \equiv t_q$. Note that based on this definition, an assignment always conflicts with itself.* □

This definition captures the influence of making an assignment on other assignments. In the following discussions, this influence is referred to as *inter-task resource constraints*, which are introduced by the constraints in Equation 5.2. Note that not only robots, but also tasks, are considered as *resources* in this definition, since once a task is assigned, it cannot be re-assigned. As with Shehory [Shehory and Kraus, 1998], non-super-additive environments are assumed so that one can restrict the maximum size of the coalitions to be $k$. For each assignment, a measure is computed to reflect the potential loss of utility due to conflicts with other assignments. This measure is then used to offset the utility of the assignment in consideration to produce the measure used at every greedy step.

### 5.3.3 *ResourceCentric*

At the beginning of any greedy step $\lambda$, $R_\lambda$ and $T_\lambda$ are used to represent the remaining sets of robots and tasks, respectively. The new heuristic, called *ResourceCentric*, chooses the assignment $m_{xy}$ in $(R_\lambda, T_\lambda)$[6] to maximize $\rho_{xy}$ (defined as follows), until no more assignments in $(R_\lambda, T_\lambda)$ exist:

$$\rho_{xy} = U(m_{xy}) - \sum_{m_{jl} \in M_{xy}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \tag{5.8}$$

in which $M_{jl}(\lambda)$ represents the set of assignments conflicting with $m_{jl}$ in $(R_\lambda, T_\lambda)$ (note that $m_{jl} \in M_{jl}(\lambda)$, given that $m_{jl} \in (R_\lambda, T_\lambda)$). Working on $(R_\lambda, T_\lambda)$ instead of $(R, T)$ ensures that new assignments do not conflict with ones that are previously chosen.

[6] An assignment $m_{xy}$ is in $(R_\lambda, T_\lambda)$ if $c_x \subseteq R_\lambda$ and $t_y \in T_\lambda$, denoted by $m_{xy} \in (R_\lambda, T_\lambda)$.

First of all, at greedy step $\lambda$, since previous assignments are already made, optimization can only be performed on $(R_\lambda, T_\lambda)$ by assuming it as a subproblem for $(R, T)$. For any assignment $m_{pq}$ in $(R_\lambda, T_\lambda)$, it is not difficult to conclude that at least one assignment would be chosen in $M_{pq}(\lambda)$ in $S^*(R_\lambda, T_\lambda)$[7]. This is because if no assignment in $M_{pq}(\lambda)$ is chosen in $S^*(R_\lambda, T_\lambda)$, it must hold that $m_{pq}$ does not conflict with any assignments in $S^*(R_\lambda, T_\lambda)$, since all conflicting assignments are in $M_{pq}(\lambda)$. As a result, $m_{pq}$ can be chosen to increase the overall utility for $(R_\lambda, T_\lambda)$, which leads to a contradiction as $m_{pq} \in M_{pq}(\lambda)$. Without prior knowledge of the optimal solution and hence assuming that all assignments are equally likely to be chosen, every assignment in $M_{pq}(\lambda)$ at least has a probability of $\frac{1}{|M_{pq}(\lambda)|}$ to be in $S^*(R_\lambda, T_\lambda)$. This holds in particular for $m_{pq} \in M_{pq}(\lambda)$. As choosing $m_{xy}$ would exclude all assignments in $M_{xy}(\lambda)$ from further consideration, hence comes the subtraction term in Equation 5.8.

**Lemma 5.3.1.** *For two steps $\lambda$ and $\gamma$ ($\lambda$, $\gamma \in \mathbb{Z}^+$) in the greedy process, given any assignment $m_{jl}$ and that $\lambda \leq \gamma$, it holds that $|M_{jl}(\lambda)| \geq |M_{jl}(\gamma)|$.*

*Proof.* First of all, given that $\lambda \leq \gamma$, it holds that $R_\gamma \subseteq R_\lambda$ and $T_\gamma \subseteq T_\lambda$. As a result, for any assignment, the number of conflicting assignments with it in step $\gamma$ cannot be greater than that in step $\lambda$. This is due to the fact that any assignment in $(R_\gamma, T_\gamma)$ would also be in $(R_\lambda, T_\lambda)$. Hence, it holds that $|M_{jl}(\lambda)| \geq |M_{jl}(\gamma)|$. □

Lemma 5.3.1 establishes the relationships between the scalars (i.e., $|M_{jl}(\lambda)|$) in Equation 5.8 in different steps of the greedy process. The following lemma provides a way to connect the solution returned by the greedy process to the optimal solution.

**Lemma 5.3.2.** *Let $G$ represent the set of assignments returned by ResourceCentric, $m^\lambda$ represent the assignment chosen at step $\lambda$, and $\mathscr{M}$ represent all assignments. The following holds:*

$$\sum_{m^\lambda \in G} \sum_{m_{jl} \in M^\lambda(\lambda)} \boldsymbol{f}(m_{jl}) = \sum_{m_{jl} \in \mathscr{M}} \boldsymbol{f}(m_{jl}) \tag{5.9}$$

[7] $S^*(I)$ is overloaded henceforth to also denote the set of chosen assignments in the optimal solution for $I$ when there is no ambiguity.

*in which $M^{\lambda}(\lambda)$ represents the set of assignments conflicting with $m^{\lambda}$ in $(R_{\lambda}, T_{\lambda})$, and $\boldsymbol{f}(m_{jl})$ represents any function that is only dependent on $m_{jl}$ given the problem instance.*

*Proof.* At any step $\lambda$, first note that since $M^{\lambda}(\lambda)$ includes all assignments that conflict with $m^{\lambda}$ in the remaining problem of $(R_{\lambda}, T_{\lambda})$, assignments in $M^{\lambda}(\lambda)$ are removed from consideration after $m^{\lambda}$ is chosen. Furthermore, *ResourceCentric* terminates when no more assignments that do not conflict with the previously chosen assignments exist. As a result, *ResourceCentric* has to remove all assignments in $\mathscr{M}$ when it terminates, since otherwise it can at least add one of the remaining assignments to the chosen set. Moreover, once an assignment is removed from consideration at $\lambda$, it would not appear again in the later steps since it conflicts with $m^{\lambda}$. Hence, every term appearing on the right hand side also appears exactly once on the left so that the conclusion holds. □

Finally, the worst case ratio for *ResourceCentric* is established in the following theorem.

**Theorem 5.3.3.** *Applying ResourceCentric to the ST-MR-IA problem while restricting the maximum coalition size to be $k$ yields a worst case ratio of $\theta = \min(2k+2, \max_{m_{jl} \in S^*}(|M_{jl}(1)|))$, in which $S^*$ is an abbreviated notation for $S^*(R,T)$.*

*Proof.* Let us first prove the $\max_{m_{jl} \in S^*}(|M_{jl}(1)|)$ part. At any greedy step $\lambda$, *ResourceCentric* needs to check all remaining assignments (i.e., assignments in $(R_{\lambda}, T_{\lambda})$, denoted by $\mathscr{M}(\lambda)$) and chooses the one with the maximum $\rho$ value. The property of $\rho$ can be analyzed by summing it over $\mathscr{M}(\lambda)$:

$$\sum_{m_{xy} \in \mathscr{M}(\lambda)} \rho_{xy} = \sum_{m_{xy} \in \mathscr{M}(\lambda)} U(m_{xy}) - \sum_{m_{xy} \in \mathscr{M}(\lambda)} \sum_{m_{jl} \in M_{xy}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \tag{5.10}$$

in which $\sum_{m_{xy} \in \mathscr{M}(\lambda)} U(m_{xy})$ is simply the sum of the utilities of all remaining assignments. In $\sum_{m_{xy} \in \mathscr{M}(\lambda)} \sum_{m_{jl} \in M_{xy}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl})$, for any assignment $m_{pq} \in \mathscr{M}(\lambda)$, $U(m_{pq})$ appears only

when $m_{xy} \in M_{pq}(\lambda)$. Hence, $U(m_{pq})$ appears $|M_{pq}(\lambda)|$ times and the total utility contributed to Equation 5.10 is the negation of the following:

$$|M_{pq}(\lambda)| \cdot \frac{1}{|M_{pq}(\lambda)|} \cdot U(m_{pq}) = U(m_{pq}) \tag{5.11}$$

As Equation 5.11 is true for every assignment in $\mathscr{M}(\lambda)$, it can be concluded that $\sum_{m_{xy} \in \mathscr{M}(\lambda)} \rho_{xy} = 0$ in Equation 5.10. As a result, in any greedy step, it can be inferred that at least one of the remaining assignments has a non-negative $\rho$ value and consequently, the assignment that is chosen by *ResourceCentric* has a non-negative $\rho$ value as it maximizes $\rho$. Based on this conclusion, it holds that $\forall\lambda$:

$$U(m^{\lambda}) \geq \sum_{m_{jl} \in M^{\lambda}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \tag{5.12}$$

The solution returned by *ResourceCentric* (denoted by $S^{RC}(R,T)$), is simply the summation of all greedy assignments in $G$:

$$\begin{aligned}
S^{RC}(R,T) &= \sum_{m^{\lambda} \in G} U(m^{\lambda}) \\
&\geq \sum_{m^{\lambda} \in G} \sum_{m_{jl} \in M^{\lambda}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) && \text{(Equation 5.12)} \\
&\geq \sum_{m^{\lambda} \in G} \sum_{m_{jl} \in M^{\lambda}(\lambda)} \frac{1}{|M_{jl}(1)|} \cdot U(m_{jl}) && \text{(Lemma 5.3.1)} \\
&= \sum_{m_{jl} \in \mathscr{M}} \frac{1}{|M_{jl}(1)|} \cdot U(m_{jl}) && \text{(Lemma 5.3.2)} \\
&\geq \sum_{m_{jl} \in S^{*}} \frac{1}{|M_{jl}(1)|} \cdot U(m_{jl}) \\
&\geq \frac{1}{\max_{m_{jl} \in S^{*}}(|M_{jl}(1)|)} \cdot S^{*}(R,T) && (5.13)
\end{aligned}$$

Let us now prove the $2k+2$ part. At step $\lambda$, $M^*(\lambda)$ is used to denote the set of assignments of $\{m \mid m \in S^*, m \in \mathscr{M}(\lambda)$ and $m$ conflicts with $m^\lambda\}$. As the size of any coalition is bounded by $k$, the following holds:

$$|M^*(\lambda)| \leq k+1 \tag{5.14}$$

Furthermore, according to the greedy criterion, it holds that $\forall\lambda\ \forall m_{pq} \in M^*(\lambda)$:

$$\begin{aligned} & U(m^\lambda) - \sum_{m_{jl} \in M^\lambda(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \\ & \geq U(m_{pq}) - \sum_{m_{jl} \in M_{pq}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \end{aligned} \tag{5.15}$$

Summing over all assignments in $M^*(\lambda)$ and then summing over all greedy steps $\lambda$ on the right hand side, it holds that:

$$\begin{aligned} & \sum_{\lambda} |M^*(\lambda)| \cdot (U(m^\lambda) - \sum_{m_{jl} \in M^\lambda(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl})) \\ & \geq \sum_{\lambda} \sum_{m_{pq} \in M^*(\lambda)} (U(m_{pq}) - \sum_{m_{jl} \in M_{pq}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl})) \end{aligned} \tag{5.16}$$

First of all, based on the definition of $M^*(\lambda)$, it holds that any assignment in the optimal solution must appear exactly once as $m_{pq}$ in $\sum_\lambda \sum_{m_{pq} \in M^*(\lambda)}$, since every assignment in the optimal solution must conflict with the chosen assignment at some step during the greedy process in order to be removed from further consideration (whether it is chosen as the greedy choice or not). Hence, it is straight forward to conclude that:

$$\sum_{\lambda} \sum_{m_{pq} \in M^*(\lambda)} U(m_{pq}) = S^*(R,T) \tag{5.17}$$

Furthermore, since any assignment can at most conflict with $k+1$ assignments in the optimal solution, it can be concluded that any assignment can appear at most $k+1$ times as $m_{jl}$ in $\sum_{\lambda} \sum_{m_{pq} \in M^*(\lambda)} \sum_{m_{jl} \in M_{pq}(\lambda)}$. Moreover, it holds that any assignment conflicting with $m^{\gamma}$ does not appear after $m^{\gamma}$ is chosen at greedy step $\gamma$. To conclude from the above, an indicator function, $\phi$, is introduced for comparing two assignments, which returns 1 only when the two are the same (0 otherwise). For any assignment $m_{xy}$ that conflicts with $m^{\gamma}$, its contribution to $\sum_{\lambda} \sum_{m_{pq} \in M^*(\lambda)} \sum_{m_{jl} \in M_{pq}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl})$ can be computed as follows:

$$\begin{aligned}
&\sum_{\lambda} \sum_{m_{pq} \in M^*(\lambda)} \sum_{m_{jl} \in M_{pq}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \cdot \phi(m_{jl}, m_{xy}) \\
&\quad = \sum_{\lambda \leq \gamma} \sum_{m_{pq} \in M^*(\lambda)} \sum_{m_{jl} \in M_{pq}(\lambda)} \frac{1}{|M_{xy}(\lambda)|} \cdot U(m_{xy}) \cdot \phi(m_{jl}, m_{xy}) \\
&\quad \leq \sum_{\lambda \leq \gamma} \sum_{m_{pq} \in M^*(\lambda)} \sum_{m_{jl} \in M_{pq}(\lambda)} \frac{1}{|M_{xy}(\gamma)|} \cdot U(m_{xy}) \cdot \phi(m_{jl}, m_{xy}) && \text{(Lemma 5.3.1)} \\
&\quad = \frac{1}{|M_{xy}(\gamma)|} \cdot U(m_{xy}) \cdot \sum_{\lambda \leq \gamma} \sum_{m_{pq} \in M^*(\lambda)} \sum_{m_{jl} \in M_{pq}(\lambda)} \phi(m_{jl}, m_{xy}) \\
&\quad \leq (k+1) \cdot \frac{1}{|M_{xy}(\gamma)|} \cdot U(m_{xy}) && (5.18)
\end{aligned}$$

Summing over all assignments removed at step $\gamma$ and then summing over all steps $\gamma$ on the right hand side (in such a way, all assignments in $\mathscr{M}$ are included exactly once, so that the indicator function on the left can be removed) gives:

$$\sum_{\lambda} \sum_{m_{pq} \in M^*(\lambda)} \sum_{m_{jl} \in M_{pq}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl})$$
$$\leq (k+1) \cdot \sum_{\gamma} \sum_{m_{xy} \in M^{\gamma}(\gamma)} \frac{1}{|M_{xy}(\gamma)|} \cdot U(m_{xy})$$
$$= (k+1) \cdot \sum_{\lambda} \sum_{m_{jl} \in M^{\lambda}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \tag{5.19}$$

Incorporating Equations 5.17 and 5.19 into Equation 5.16, it holds that:

$$\sum_{\lambda} |M^*(\lambda)| \cdot (U(m^{\lambda}) - \sum_{m_{jl} \in M^{\lambda}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}))$$
$$\geq S^*(R,T) - (k+1) \cdot \sum_{\lambda} \sum_{m_{jl} \in M^{\lambda}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \tag{5.20}$$

Given that $|M^*(\lambda)| \geq 0$ and Equation 5.14, it can also be concluded that:

$$\sum_{\lambda} |M^*(\lambda)| \cdot (U(m^{\lambda}) - \sum_{m_{jl} \in M^{\lambda}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}))$$
$$\leq \sum_{\lambda} |M^*(\lambda)| \cdot U(m^{\lambda})$$
$$\leq (k+1) \cdot S^{RC}(R,T) \tag{5.21}$$

Combining Equations 5.20 and 5.21, it can be concluded that:

$$
\begin{aligned}
&(k+1) \cdot S^{RC}(R,T) \\
&\quad \geq S^*(R,T) - (k+1) \cdot \sum_{\lambda} \sum_{m_{jl} \in M^{\lambda}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \\
&\quad \geq S^*(R,T) - (k+1) \cdot S^{RC}(R,T) \qquad \text{(First inequality in Equation 5.13)}
\end{aligned}
$$

And finally, it can be concluded that:

$$S^*(R,T) \leq (2k+2) \cdot S^{RC}(R,T) \tag{5.22}$$

Hence the conclusion holds. □

For problem instances in which assignments in the optimal solution conflict less with other assignments, the $\max_{m_{jl} \in S^*}(|M_{jl}(1)|)$ ratio guarantees a good quality solution. This is especially true in multi-robot systems with heterogeneous robots, since different robots can only handle specific tasks. Otherwise, the solution quality is bounded by $2k + 2$. Although this worst case ratio is slightly worse than *AverageUtility*, it is more difficult to satisfy the boundary conditions (i.e., all inequalities in the proof hold as equalities simultaneously), which may suggest that *ResourceCentric* would perform well on average even when $\max_{m_{jl} \in S^*}(|M_{jl}(1)|)$ is large.

### 5.3.4 The algorithm of *ResourceCentric*

Algorithm 8 presents an implementation of *ResourceCentric* using a graph structure. The algorithm starts with building the graph, which has a node for each assignment and an edge between two nodes if they conflict with each other. Then the algorithm proceeds with the greedy iterations and assigns a task at each step. For every assignment made, the assignment node and all assignments

connecting with it, as well as all connecting edges[8], are removed from the graph. This process continues until all assignments are removed (i.e., the graph becomes empty).

The complexity for creating the graph is bounded by $O(|T||C||\mathscr{M}|)$. Each greedy step is bounded by $O(|\mathscr{M}|^2)$[9]. As there can at most be $\min(|R|, |T|)$ assignments, the complexity for the entire process is bounded by $O(\min(|R|, |T|) \cdot |T|^2|C|^2)$ (note that $|\mathscr{M}|$ is bounded by $|T||C|$).

### 5.3.5 *ResourceCentricApprox*

One problem with *ResourceCentric*, however, is the computational complexity. Although *ResourceCentric* runs in polynomial time with respect to $|C|$, as discussed, $|C|$ can grow exponentially with $|R|$. When the number of robots in the distributed system is large, *ResourceCentric* can be significantly slower than *AverageUtility* and *MaxUtility*, which run in $O(\min(|R|, |T|) \cdot |T||C|)$. Instead of computing $\rho$ exactly, the following approximation can be used at greedy step $\lambda$:

$$
\begin{aligned}
\hat{\rho}_{xy} &= U(m_{xy}) - \sum_{m_{jl} \in M_{xy}(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) \\
&\approx U(m_{xy}) - \sum_{r_i \in c_x} \sum_{m_{jl} \in M_i(\lambda)} \frac{1}{|M_{jl}(\lambda)|} \cdot U(m_{jl}) && \text{(Break onto each robot)} \\
&= U(m_{xy}) - \sum_{r_i \in c_x} \overline{\frac{|M_i(\lambda)|}{|M_{jl}(\lambda)|} \cdot U(m_{jl})}_{m_{jl} \in M_i(\lambda)} \\
&\approx U(m_{xy}) - \sum_{r_i \in c_x} \overline{\frac{|M_{i,l}(\lambda)|}{|M_l(\lambda)|} \cdot U(m_{jl})}_{m_{jl} \in M_i(\lambda)} && \text{(Remove dependency on } j\text{)} \\
&= U(m_{xy}) - \sum_{r_i \in c_x} \overline{\theta_{il}(\lambda) \cdot U(m_{jl})}_{m_{jl} \in M_i(\lambda)} && (5.23)
\end{aligned}
$$

in which the bars over the formulas represent averaging operations, $M_i(\lambda)$ represents the set of assignments in $(R_\lambda, T_\lambda)$ that rely on $r_i$ (i.e., $r_i$ is in the coalitions for the assignments), $M_{i,l}(\lambda)$

[8] In our implementation of *ResourceCentric*, the edge information is not kept in the node structure in order to reduce the space complexity.

[9] Note that computing $\frac{1}{|M_u|}$ in the inner loop requires only constant time in our algorithm. For each node, this count is computed when the graph is initially created and kept updated when a chosen assignment and its neighbors are removed from the graph.

**Algorithm 8** *ResourceCentric*

Generate the set of coalitions $C$, with maximum size $k$.
Create an undirected graph $G : (V, E)$.
**for all** $t_l$ in $T$ **do**
  **for all** $c_j$ in $C$ **do**
    **if** $c_j$ satisfies $t_l$ **then**
      Create a node $m_{jl}$.
      Compute $U(m_{jl})$.
      **for all** $v \in V$ **do**
        **if** $v$ conflicts with $m_{jl}$ **then**
          Connect $v$ and $m_{jl}$.
        **end if**
      **end for**
    **end if**
  **end for**
**end for**
**while** $G$ is not empty **do**
  **for all** $v$ in $V$ **do**
    **for all** $u$: $u$ and $v$ are connected **do**
      Compute $\frac{1}{|M_u|} \cdot U(u)$
    **end for**
    Compute $\rho_v$.
  **end for**
  Choose $v^{RC}$ that maximizes $\rho$.
  Remove $v^{RC}$, its neighbors and edges connecting these nodes with the remaining nodes from $G$.
**end while**
**return** The chosen nodes (i.e., assignments).

represents the set of assignments in $(R_\lambda, T_\lambda)$ for task $t_l$ relying on $r_i$, and $M_l(\lambda)$ represents the set of assignments in $(R_\lambda, T_\lambda)$ for task $t_l$. The $\approx$ sign in the above formula should be interpreted as "is approximated by". Note that in the first approximation, the same assignments may appear more than once in $\sum_{r_i \in c_x} \sum_{m_{jl} \in M_i(\lambda)}$. The second approximation is introduced to avoid nested loops for iterating through all coalitions (each loop is on the order of $O(|C|)$), so that the computational complexity can be reduced.

At each step $\lambda$, for each remaining task $t_l$, compute $\theta_{il}(\lambda)$ for each remaining $r_i$, which is a measure that reflects how much $t_l$ relies on $r_i$. When choosing an assignment $m_{xy}$, for each robot $r_i \in c_x$, first, compute the expected loss of utility due to the assignment of $r_i$ as:

$$E_\lambda(r_i) = \overline{\theta_{il}(\lambda) \cdot U(m_{jl})}_{m_{jl} \in M_i(\lambda)} \tag{5.24}$$

Afterwards, compute $\hat{\rho}_{xy} = U(m_{xy}) - \sum_{r_i \in c_x} E_\lambda(r_i)$ and choose the assignment that maximizes it. This heuristic is referred to as *ResourceCentricApprox*.

Now let's trace back to our motivational example at the beginning of this section. Suppose that the only three robots that have the first three capabilities are $r_1$, $r_2$, $r_3$ with capability vectors $\boldsymbol{B}_1 = (1, 0, 0, 0, 0)$, $\boldsymbol{B}_2 = (0, 1, 0, 0, 0)$, $\boldsymbol{B}_3 = (0, 0, 1, 0, 0)$. In this case, it holds that $\theta_{12} = \theta_{23} = \theta_{34} = 1.0$, since robot $r_1$, $r_2$ and $r_3$ are prerequisites for tasks $t_2$, $t_3$ and $t_4$, respectively. As a result, when assigning $t_1$ with a slightly higher reward, the value of $\hat{\rho}$ for the assignment of $\{r_1, r_2, r_3\} \rightarrow t_1$ would still be lower than for the other tasks and hence $t_1$ would not be chosen. Although this may seem to be an arbitrary example, it is shown in the result sections that *ResourceCentric* and *ResourceCentricApprox* actually perform better, which indicates that similar situations often occur in random configurations.

Table 5.1: Summary of discussed methods with maximum coalition size $k$

| **Name** | **Formulation** | **Worst Case Ratio** | **Amort. Worst Case Time** |
|---|---|---|---|
| *AverageUtility* | Theorem 5.2.1 | $2k$ | $O(\min(\|R\|, \|T\|) \cdot \|T\|\|C\|)$ |
| *MaxUtility* | Theorem 5.2.2 | $k+1$ | $O(\min(\|R\|, \|T\|) \cdot \|T\|\|C\|)$ |
| *ResourceCentric* | Equation 5.8 | $\min(2k+2,$ $\max_{m_{jl} \in S^*}(\|M_{jl}(1)\|))$ | $O(\min(\|R\|, \|T\|) \cdot \|T\|^2\|C\|^2)$ |
| *ResourceCentricApprox* | Equation 5.23 | Not Determined | $O(\min(\|R\|, \|T\|) \cdot \|R\|\|T\|^2\|C\|)$ |

### 5.3.6 The algorithm of *ResourceCentricApprox*

One way to implement *ResourceCentricApprox* is presented in Algorithm 9. The algorithm starts with creating a hash table for each task and a hash table for each robot and task pair. After it fills the hash tables, the algorithm proceeds with the greedy iterations to make assignments.

The complexity for creating the hash tables is bounded by $O(|R||T|)$. The complexity for filling the hash tables is $O(|T||C|)$. Each greedy choice requires $O(|R||T|^2|C|)$ computations in the worst case. Hence, the computational complexity for this implementation is bounded by $O(\min(|R|, |T|) \cdot |R||T|^2|C|)$. Hence, it can be concluded that this algorithm performs almost as well as *AverageUtility* and *MaxUtility* in terms of worst case running time.

Table 5.1 provides a summary of all methods that have been discussed in this research, including their formulations, worst case ratios, and amortized worst case running times.

## 5.4 Extended formulation

One issue with the formulation of ST-MR-IA is that it is insufficient for complicated scenarios. Hence, in this section, the formulation of the ST-MR-IA problem is extended to incorporate general task dependencies. This extended formulation is referred to as the ST-MR-IA-TD problem. A result on the hardness of approximating ST-MR-IA-TD is provided afterwards. An algorithm that utilizes the discussed methods for ST-MR-IA to address this extended formulation of the problem is also provided.

**Algorithm 9** *ResourceCentricApprox*

Generate the set of coalitions $C$, with maximum size $k$
**for all** $t_l$ in $T$ **do**
  Create a hash table $H_l$
  **for all** $i$ in $R$ **do**
    Create a hash table $H_{i,l}$
  **end for**
**end for**
**for all** $t_l$ in $T$ **do**
  **for all** $c_j$ in $C$ **do**
    **if** $c_j$ satisfies $t_l$ **then**
      Add $c_j$ into $H_l$
      **for all** $r_i$ in $c_j$ **do**
        Add $c_j$ into $H_{i,l}$.
      **end for**
    **end if**
  **end for**
**end for**
**while** $H$ tables are not all empty **do**
  **for all** $r_i$ in remaining $R$ **do**
    **for all** $t_l$ in remaining $T$ **do**
      **for all** $c_j$: $r_i \in c_j$ **do**
        Compute $\frac{|H_{i,l}|}{|H_l|} \cdot U(m_{jl})$.
      **end for**
    **end for**
    Compute $E(r_i)$.
  **end for**
  **for all** $m_{xy}$ remaining **do**
    Compute $\hat{\rho}_{xy} = U(m_{xy}) - \sum_{r_i \in c_x} E(r_i)$.
  **end for**
  Choose $m^{RCA}$ that maximizes $\hat{\rho}$.
  **for all** $r_i$ in $c^{RCA}$ **do**
    **for all** $t_l$ in $T$ **do**
      **for all** $c$ in $H_{i,l}$ **do**
        Remove $c$ from all tables.
      **end for**
    **end for**
  **end for**
  Clear tables involving $t^{RCA}$.
**end while**

### 5.4.1 Adding task dependencies

As discussed, one issue with the formulation of ST-MR-IA is that it does not incorporate task dependencies, which can be critical for real world applications. Previous approaches (e.g., [Shehory and Kraus, 1998]) have studied precedence ordering between tasks, in which one task can only be assigned when other tasks in its precedence order are also assigned. For example, in the disaster response scenario that is presented earlier, the task for addressing the fires in buildings is not possible if the roads to the buildings are not cleared. However, the definition of precedence order should be extended to incorporate scenarios in which assigning other tasks facilitates the execution of one task, instead of being a prerequisite for the task. For example, although there might exist an alternative road that is not blocked, taking this alternative route can potentially be less optimal than clearing the blocked road first and using it to reach the buildings (e.g., it may take longer to reach the buildings via the alternative route). Note that the precedence order in [Shehory and Kraus, 1998] can then be considered as a special case of this extended formulation (i.e., a task yields a very small utility if the tasks in its precedence order are not satisfied, essentially making the task be ignored until its precedence order is satisfied). Also note that the precedence order does not necessarily have to specify a complete ordering of task execution (i.e., some tasks have to be completed before starting the execution of some others). A task and tasks in its precedence order may or may not be able to be executed simultaneously. To avoid the scheduling issue that may arise from this complexity, in this work, a similar approach as in [Shehory and Kraus, 1998] is utilized, such that resources are pre-allocated to the task and tasks in its precedence order at the same time.

Another aspect in task dependencies that has not been considered occurs when there are alternative tasks, such that the assignment of any one of them makes the others unnecessary. In the same example, when there are alternative roads that are all blocked leading to the same buildings, only one of them needs to be cleared. Correspondingly, these other tasks may still be beneficial even though one of them is assigned. For example, to achieve more efficiency, several alternative

roads may need to be cleared (i.e., for other traffic coming in and out without interfering with the truck agents). To incorporate these considerations, the following component is added into the formulation of ST-MR-IA:

- a set of task dependencies $\Gamma$. Each dependency for a task $t$ is defined as a pair $\tau = (\mathscr{T}, \Re^+)$, in which $\mathscr{T} \subseteq \{T - t\}$ and $\mathscr{T} \neq \emptyset$. The real positive value (denoted by $v_D$) is the updated reward of $t$ when this dependency is satisfied.

For example, for specifying a linear ordering between $t_1$, $t_2$ and $t_3$ such that $t_1$ must be satisfied before $t_2$ and $t_2$ must be satisfied before $t_3$, a task dependency $(\{t_1\}, v_D)$ for $t_2$ and a task dependency $(\{t_2\}, v'_D)$ for $t_3$ need to be defined. This extended formulation of the ST-MR-IA problem is denoted as ST-MR-IA-TD. Given a task $t_l$, precedence orders can then be implemented by requiring that $v_D \geq \boldsymbol{V}[l]$; alternative tasks can be implemented by requiring that $v_D < \boldsymbol{V}[l]$. Dependencies of these two aspects are considered separately in the algorithm that is presented at the end of this section. This is due to the fact that when $v_D \geq \boldsymbol{V}[l]$, tasks in the dependency are desirable; on the other hand, when $v_D < \boldsymbol{V}[l]$, tasks in the dependency should generally be avoided (when $v_D$ for the task is so small such that the utility for any assignment is non-positive, the task would effectively be ignored). In cases when a task has multiple dependencies, rules should be defined for cases when multiple dependencies are satisfied simultaneously. This aspect will be addressed in more depth in our future work.

### 5.4.2 Problem analysis

In this section, the result on the hardness of approximating the ST-MR-IA-TD problem is provided.

**Theorem 5.4.1.** *It is NP-hard to approximate the ST-MR-IA-TD problem with a poly-time worst case ratio that is independent of $v_D$ values in task dependencies.*[1]

[1]Recall that a poly-time worst case ratio is a worst case ratio that can be computed in polynomial time given the problem instance.

*Proof.* The proof is by contradiction. Let's suppose that a polynomial time approximation algorithm, *TD_Approx*, does exist with a poly-time worst case ratio of $\theta$. Next, it can be shown that algorithm *TD_Approx* can be utilized to solve the *3-Partition* problem, which is strongly NP-complete. Any instance of the *3-Partition* problem can be represented as *3-Partition*$(S, M$: $S = \{e_1, e_2, ..., e_{3M}\}, e_j \in \mathbb{Z}^+)$. Let the sum of all elements in $S$ be $M \cdot B$. The problem is to determine whether or not the set $S$ can be divided into $M$ sets such that the sum of the elements in each set is $B$. The problem remains NP-complete even when one requires that $\forall e_j \in S : \frac{B}{4} < e_j < \frac{B}{2}$. These constraints imply that if a solution exists for the *3-Partition* problem, each set would have exactly 3 elements.

Next, it is shown that an instance of *ST-MR-IA-TD(R, C, T,* $\boldsymbol{W}$*,* $\boldsymbol{V}$*, Cost,* $\Gamma$*)* can be constructed from an instance of *3-Partition*$(S, M)$. First of all, for each element $e_j \in S$, construct a robot $r_j$ which has a *1-D* (i.e., $H = 1$) capability vector with value equal to the integer value of the element $e_j$. Then, the set $C$ of coalitions can be created by including all coalitions with exactly 3 robots. Note the size of $|C|$ is $\binom{3M}{3}$, which is polynomial in $M$. This is important for guaranteeing the validity of the transformation. On the other hand, this does not influence the determination of the existence of a solution, as each set would have exactly 3 elements anyway.

Create $M$ tasks to be accomplished, so that $T = \{t_1, t_2, ..., t_M\}$. Each $t_l$ has a capability requirement of exactly $B$. Since $H = 1$, the cost vector $\boldsymbol{W}$ reduces to a scalar $w$, which is set to 0 for simplicity. The initial rewards for all tasks are assigned to be 1. Assume that there is no communication cost so that the function *Cost* invariably returns 0. Finally, for $\Gamma$, a task dependency is defined, $(\{T - t_M\}, v_D)$, for $t_M$. Since $\theta$ is assumed to be computable in polynomial time and independent of $v_D$ values, $\theta$ can be computed and assign $v_D$ to be $M \cdot \theta - (M - 1)$. An instance of the ST-MR-IA-TD problem is thus constructed from an instance of the *3-Partition* problem.

Now the *3-Partition* problem can be solved using *TD_Approx* as follows.

$$\textit{3-Partition} = \begin{cases} 1 & \textbf{if } TD\_Approx \geq M \\ 0 & \textbf{if } otherwise \end{cases} \tag{5.25}$$

Whenever *TD_Approx* returns an overall utility of no less than $M$, it holds that all $M$ tasks must be in the solution. This is because accomplishing less than $M$ would receive an overall utility no more than $M-1$. Hence, it holds that there exists a way in the ST-MR-IA-TD problem to allocate exactly 3 robots to each task. This solution is clearly also a solution for the *3-Partition* problem. On the other hand, if *TD_Approx* returns an overall utility of less than $M$, it can be concluded that the optimal solution must achieve an overall utility of less than $M \cdot \theta$, according to the definition of worst case ratio. If there exists a solution for the *3-Partition* problem, one can apply the solution to the ST-MR-IA-TD problem so that all tasks are included. The corresponding overall utility received is then $M \cdot \theta - (M-1) + (M-1) = M \cdot \theta$. This contradicts with the previous conclusion that the optimal solution must achieve an overall utility of no more than $M \cdot \theta$. Hence, there could not exist a solution for the *3-Partition* problem. In such a way, the *3-Partition* problem can be solved. Unless $P \equiv NP$, the conclusion holds. □

### 5.4.3 Allocation with task dependencies

For addressing the ST-MR-IA-TD problem, the greedy approach used in [Shehory and Kraus, 1998] is adapted for task allocation with precedence orders. However, since task dependencies can either increase or decrease the reward, the two cases are considered separately. For any task $t_l$ with a set of dependencies $\Gamma_l \subseteq \Gamma$, $\Gamma_l$ can be separated into two disjoint sets:

- $\Gamma^p$: the set of dependencies that satisfies $v_D \geq \boldsymbol{V}[l]$.
- $\Gamma^r$: the set of dependencies that satisfies $v_D < \boldsymbol{V}[l]$.

$\Gamma^p$ can be considered as the set of precedence orders in [Shehory and Kraus, 1998]. As in [Shehory and Kraus, 1998], at each greedy step, instead of making a single assignment at a time, the algorithm checks each task with tasks in each of its precedence orders (i.e., dependencies in $\Gamma^p$ for

the task). For each set of tasks, a set of assignments are made using one of the previously discussed methods for the ST-MR-IA problem. The algorithm then chooses the set of assignments that gives the best *quality measure*[2] (similar to $p$-value in [Shehory and Kraus, 1998]). The difference from [Shehory and Kraus, 1998] is that task dependencies in $\Gamma^r$ are also considered in the computation of the *quality measure* for each set of assignments to incorporate their influences. Note that although one cannot directly compare this approach with [Shehory and Kraus, 1998] due to the different measures used (i.e., cost measures in [Shehory and Kraus, 1998] rather than utility measures in our work) and that $\Gamma^r$ is also considered, the algorithm in Section 5.4.4 using *AverageUtility* is in fact very similar to that in [Shehory and Kraus, 1998] when ignoring these differences.

To simplify the rules when multiple dependencies are satisfied simultaneously, for each task $t_l$, let all dependencies in $\Gamma^p$ assume the same value $v_D^p$ and all dependencies in $\Gamma^r$ assume $v_D^r$. When only dependencies in $\Gamma^p$ are satisfied, the reward of the task is updated to be $v_D^p$; when only dependencies in $\Gamma^r$ are satisfied, the reward is updated to be $v_D^r$; when there are satisfied dependencies in both sets, the reward is updated to be $v_D^r$. This set of rules is reasonable when tasks tend to have few dependencies (i.e., at most 1 in either set) or dependencies from the same set have similar effects on the task. More complicated rules can be designed without influencing the following discussions. The influences of task dependencies on the task rewards are computed according to these rules.

### 5.4.4 The algorithm for task allocation with task dependencies

The algorithm to address the ST-MR-IA-TD problem is shown in Algorithm 10. At every step, for every remaining task, for every task dependency in $\Gamma^p$ for the task, the algorithm creates a set of tasks that includes the task along with all other tasks in the dependency. This set of tasks is fed to one of the methods for addressing the ST-MR-IA problem. After the assignments are made, the

[2]The *quality measure* for a set of assignments is computed as the combination of the measures, used by the chosen method for making these assignments (e.g., $\rho$ for *ResourceCentric*), while incorporating the influences of task dependencies. For example, when using *MaxUtility*, the *quality measure* is computed as the summation of the utility measures for these assignments, considering the change of task rewards due to the satisfied task dependencies.

*quality measure* for this set of assignments is evaluated. Note that all dependencies (i.e., both $\Gamma^p$ and $\Gamma^r$) of the previously chosen tasks and tasks in these assignments are checked to compute the influence (i.e., due to the updates of task rewards) on the *quality measure* due to newly satisfied dependencies. The set of assignments with the best *quality measure* is then chosen.

**Algorithm 10** Task allocation with task dependencies

**while** remaining tasks can still be assigned **do**
  **for all** $t_l$ in $T$ **do**
    **for all** $\tau$ in $\Gamma^p$ for $t_l$ **do**
      Create the set of tasks to include $t_l$ and tasks in $\tau$ .
      Invoke a method (e.g., *AverageUtility, ...*) to choose assignments for this set of tasks.
      Record the chosen assignments as $M_{l\tau}$.
      **if** $M_{l\tau} \neq \emptyset$ **then**
        **for all** $m_{jl} \in M_{l\tau}$ **do**
          Compute the measure for the greedy choice based on the chosen method.
          Compute the influence of newly satisfied dependencies (based on $\Gamma^p$ and $\Gamma^r$ of tasks in the chosen assignments, including $t_l$) as a result of choosing $m_{jl}$.
          Incorporate the influence into the measure for the greedy choice.
          Assume that $m_{jl}$ is chosen for the next iteration.
        **end for**
        Combine the measures for all $m_{jl} \in M_{l\tau}$ as the *quality measure*.
      **end if**
    **end for**
  **end for**
  Choose the $M^*_{l\tau}$ with the maximum *quality measure*.
  Remove the assigned robots and tasks in $M^*_{l\tau}$ from $R$ and $T$.
**end while**

# Chapter 6

# Task allocation with executable coalitions

In this chapter, a framework [Zhang and Parker, 2012d] for combining the previous approaches is presented, in which a layering technique is used. For achieving multi-robot tasks, furthermore, several additional issues must also be considered, including reducing the complexity of task allocation and addressing the situations when no executable coalitions exist for tasks. To the best of our knowledge, this is the first work that addresses the task allocation problem with executable coalitions to reduce the problem complexity with multi-robot tasks. Furthermore, a new approach is introduced that can search for different ways to satisfy preconditions of the required tasks based on the current situations and autonomously create partial order plans. The implementation of this approach uses the reasoning process of IQ-ASyMTRe and is integrated with it in a natural way. In the following, how to layer task allocation with IQ-ASyMTRe is first discussed, and then the process to address tasks with no executable coalitions is presented. Simulation results are presented in Section 7.5. The method in this chapter aims at providing a starting point to integrate the previous aspects (discussed in this research) in one framework to achieve system autonomy for multi-robot tasks.

## 6.1 Task allocation with IQ-ASyMTRe

Next, the layering technique is discussed, as well as addressing tasks with no executable coalitions using a market-based approach. Specific algorithms are presented afterwards.

### 6.1.1 Layering IQ-ASyMTRe with task allocation

Tasks can be represented as the required behaviors (i.e., MSs) to be activated. IQ-ASyMTRe creates coalitions for tasks when the required information for the behaviors can be retrieved. Since the reference of information is complete, IQ-ASyMTRe guarantees forming executable coalitions. Task allocation can be implemented similarly as in [Tang and Parker, 2007] based on a marker-based approach. An auctioneer (i.e., the central task allocation process) announces tasks to the robots, while the robots (running IQ-ASyMTRe algorithms) reason about possible coalitions and submit bids for tasks. The auctioneer (also running a task allocation algorithm) then determines the winner coalitions for the tasks and assigns them to these coalitions. A coalition (and a robot) can only win one bid at a time. Robustness can be achieved by setting timers for certain events (e.g., when no bids for a task are received after a period of time, the task can be re-announced).

#### Interface with task allocation

To interface with the task allocation methods, information about the robots in the coalitions, the costs of coalitions, rewards for the tasks, and precedence orders between tasks must be included in the bids. One assumption made is that capabilities are not shared between different coalitions. This is almost always true in multi-robot systems (unlike in multi-agent systems), since capabilities are not transferable. As a result, information regarding the capabilities allocated by the coalitions does not need to be provided. This reduces the complexity of the central task allocation process.

In IQ-ASyMTRe, coalition members include robots that provide the necessary information and the robots that execute the required behaviors. For example, in Figure 4.2, while $R_1$ is the robot that executes the behavior (i.e., the MS), $R_2$ provides the required information to $R_1$. Hence, both

$R_1$ and $R_2$ are in the coalition. Coalitions with a single robot can be created when individual robots can execute the desired behaviors without help. Rewards of tasks and precedence orders between tasks are often specified a priori. Next, how the costs of coalitions are computed is discussed.

**Coalition cost**

The cost of a coalition should be computed to approximate the actual cost for the coalition to accomplish the task. In IQ-ASyMTRe, the costs of the (sensory and computational) capabilities can be computed as the summation of the costs of the MSs, PSs and ESs activated; similarly, communication and coordination costs can be computed based on the costs of the CSs used. When execution times can be estimated, they can be incorporated by defining the costs of schemas to be unit-time costs. Furthermore, to consider the influence of information quality, the *coalition quality measure*[1] is associated with the success ratio ($\theta$) of the coalition using a task-specific function, $F : [0, 1] \times T \rightarrow [0, 1]$, in which $[0, 1]$ is the space of all possible values of coalition quality and $T$ is the space of all possible types of tasks.

Note that coalition quality is computed to reflect the utility for using the information required by the coalition. While certain tasks are not influenced much by the utility of the information, others can be significantly affected. For example, in the robot navigation task, the success ratio is not influenced much by whether the follower robot is in a desirable configuration relative to the leader (e.g., close to the leader), since the robots can communicate to coordinate their actions. On the other hand, in a robot tracking task, whether the robot can successfully track the target is highly dependent on their relative configuration, since the target can move out of sight easily when the configuration is undesirable. Given that the coalition is committed to accomplish the task once assigned, the expected cost can be computed for a coalition $c$ and task $t$ as $cost(c, t) = E(\widehat{cost}(c, t)) = \widehat{cost}(c, t)/F(Q_c, Y_t)$, in which $\widehat{cost}(c, t)$ represents the summation of the costs of all activated schemas in the coalition for the task, $Q_c$ and $Y_t$ represent the coalition quality of $c$ and

[1] IQ-ASyMTRe computes the coalition quality measure by simply multiplying the related information quality measures.

the task type of $t$, respectively. When the expected costs of all executable coalitions for a task are greater than the reward of the task (i.e., all coalitions are very likely to fail given the current situation, so that executing the task would not be beneficial), the task would be handled as if no executable coalitions exist according to the process in the following section.

### 6.1.2 Tasks with no executable coalitions

One obvious advantage with forming executable coalitions is that robots actually know how to execute them to accomplish the tasks. Furthermore, since the number of executable coalitions often are much smaller than the number of feasible coalitions in the current situation, the complexity for task allocation can be significantly reduced. Consequently, however, IQ-ASyMTRe cannot address tasks for which no executable coalitions exist.

#### Extending MS

First of all, in IQ-ASyMTRe, it is noticed that the preconditions of behaviors are input information instances of the required MSs for the tasks (e.g., $F_G(local)$ in Figure 3.3). A task is not executable if the required information cannot be retrieved by any coalitions in the environment. In order to introduce executable coalitions, the robots must have the capability to obtain information that is initially not retrievable. To incorporate this capability into the general framework of schema theory, the definition of MS can be extended so that it can not only output commands for actuators, but also output information. In such a way, the IQ-ASyMTRe algorithms can remain almost unchanged. Although the introduction of this new kind of MS does not influence how MSs are used, it is referred to as IMS (Information MS) when there is necessity to distinguish it.

To use this capability, all that needs to be done is to create and assign new tasks for IMSs. However, it is unlikely that IMSs always exist to directly retrieve the required information. To address this issue, the capability of IQ-ASyMTRe can be used to reason about alternative ways to retrieve the information. An illustration example is presented in Figure 6.1 for the robot navigation task. When no potential leader is in the FOV of the follower ($R_1$), no executable coalition exists.

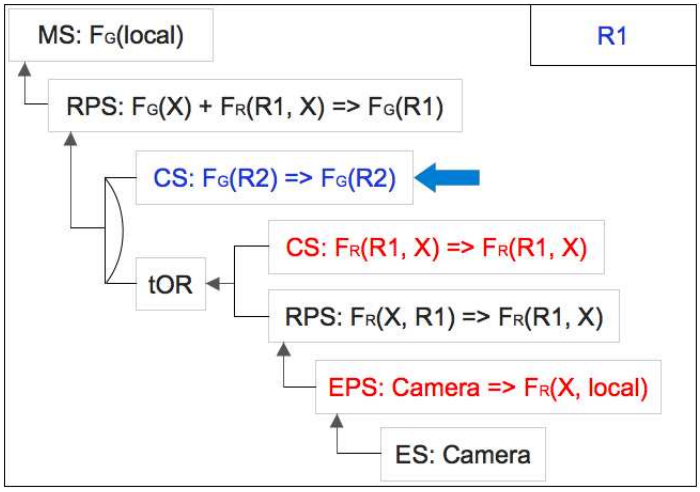



Figure 6.1: An illustration example for using IMS in the robot navigation task.

Suppose that an IMS (i.e., *find-entity*) is implemented on both $R_2$ (a potential leader) and $R_1$ to search for entities within the communication range in the environment. Given that $R_1$ knows that $R_2$ can localize (from CS: $F_G(R_2) \Rightarrow F_G(R_2)$ in blue), $R_1$ either needs to find a way to retrieve $F_R(R_1, R_2)$ or $F_R(R_2, R_1)$ (in red). While the first can be more conveniently retrieved by $R_2$ executing *find-entity*, the latter is more convenient for $R_1$.

### Information task request

Task allocation is performed in two phases. In the first phase, called the *Easy Auction* phase, robots search for executable coalitions for the broadcasted tasks and submit bids to the auctioneer. The auctioneer assigns tasks based on all submitted bids. For tasks that no bids are submitted, an additional auctioning step is initiated for IMS tasks. In this second phase (called the *IMS Auction* phase), the robots reason about the alternative ways to retrieve the input information instances as demonstrated in Figure 6.1. The robots then submit *information task requests* for requesting IMS tasks to the auctioneer. The auctioneer then considers these new tasks as preconditions[2] for

[2]Here, preconditions refer to execution orders specified for partial order plans, which differ from our previous mentioning of preconditions as input information for behaviors. References of preconditions in the remaining discussions should be unambiguous given the context.

the initiating tasks in the partial order plan. These new tasks are then auctioned using the *Easy Auction* phase. Note that these new tasks are handled in the same way and the two-phase process may apply recursively.

In this way, partial order plans can be autonomously generated. Since these IMSs are created to provide input information for the required MSs, once the new tasks are accomplished, the input information of the required MSs would be satisfied and the initiating tasks can be executed. However, caution must be taken to avoid IMS tasks removing the already satisfied input information. Detail discussion is out of the scope of this research and the issue will be addressed in future work.

Note that the creation of partial order plans unavoidably introduces scheduling issues, which greatly increase the complexity of the task allocation problem. In our current approach, these scheduling issues are ignored. For each task allocation process, tasks are considered only when all preconditions in their partial order plan are satisfied.

## 6.2 Algorithms for task allocation

**Algorithm 11** Auctioneer process

```
Create empty new_task and announced_task lists.
while true do
  Receive new tasks and put them on the new_task list.
  for all tasks in announced_list that are initiating tasks for the new IMS tasks received do
    Update the task's preconditions.
    Move the task from announced_list to new_list.
  end for
  IMS Auction: announce tasks in announced_list.
  Easy Auction: announce tasks in new_task list for which preconditions are satisfied.
  Move the announced tasks to announced_list.
  Wait a while for bids.
  Collect bids from robots.
  Invoke task allocation algorithms to determine the task assignments.
  Remove tasks that are assigned from new_task list.
  Move tasks for which no bids are submitted or no bids are beneficial to announced_list.
end while
```

**Algorithm 12** Robot process

```
while true do
  if the robot has a winning bid then
    Set up the coalition and execute the task.
  end if
  Receive new task announcements.
  for all received tasks do
    if task announced for Easy Auction then
      Invoke IQ-ASyMTRe to search for executable coalitions and submit bids.
    else if task announced for IMS Auction then
      Invoke IQ-ASyMTRe to submit information task requests.
    end if
  end for
end while
```

The algorithms for the auctioneer and robot processes are provided in Algorithms 11 and 12, respectively. The auctioneer maintains a list of new tasks and a list of announced tasks. In each task allocation process, the auctioneer announces tasks for which all preconditions in the partial order plan are satisfied. It then receives bids and allocates tasks to winner coalitions. Tasks for which no bids are submitted or no bids are beneficial are moved to the list of announced tasks. Tasks in the list of announced tasks are announced in the *IMS Auction* phase. Whenever the auctioneer receives information task requests (IMS tasks), the auctioneer updates the preconditions of the initiating tasks in the list of announced tasks and moves them to the list of new tasks. Note that new tasks are not announced until all preconditions in the partial order plan are satisfied. For the robots, when there is a winning bid, they set up coalitions to accomplish the task; otherwise, they reason about the solutions for the announced tasks.

# Chapter 7

# Experimental results

Simulations and experimental results for the approaches discussed in this dissertation are presented in this chapter, which demonstrate the novelty of these approaches. First, results for forming coalitions using IQ-ASyMTRe are presented in Section 7.1. Then, results for executing coalitions using IQ-ASyMTRe$^+$ are presented in Section 7.2. Finally, results for task allocation, task allocation with the extended formulation, and for combining the previous aspects are presented in Sections 7.3, 7.4 and 7.5, respectively.

## 7.1 Results for IQ-ASyMTRe

In this section, simulations and experimental results are provided to demonstrate the capabilities of IQ-ASyMTRe for various applications. All simulations are implemented in Player/Stage [Gerkey et al., 2003], and are run on a 2.4GHz Core 2 Duo laptop with 2GB memory; wall-clock times are reported.

Table 7.1: Information required in the navigation task

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

### 7.1.1 Simulations

#### Solution space and potential solutions

First, the solution spaces produced by IQ-ASyMTRe are shown for the navigation task, in which the goal is to activate a MS for navigation on different robots. The information instances required are listed in Table 7.1 along with brief descriptions. The potential solutions are ordered based on the costs of schemas they use, which are currently statically defined as follows: $cost$(RPS) = 0.5, $cost$(ES-EPS) = 1.0, $cost$(CS) = 2.0 and $cost$(MS) = 4.0. $F_G(goal)$ and $F_M$ are provided a priori, which incur no costs. Since $F_A(local)$ can only be retrieved using the local laser sensor, it is also ignored for conciseness. Table 7.2 lists the potential solutions for robots equipped with a fiducial, a laser and a GPS sensor; RPSs are not shown for brevity. Encoding entity information into information enables IQ-ASyMTRe to find solutions (i.e., 6-11) that are not discernible by architectures that use only information types[1]. Hence, more flexibility and completeness are achieved. Note that potential solutions requiring the same set of information instances with higher costs can be removed to reduce the search space without affecting the completeness (e.g., using a post-processing algorithm).

#### Forming executable coalitions

In this simulation, a scenario of the navigation task is presented to demonstrate the intuition behind forming executable coalitions. It is assumed that there are 3 robots without a localization

[1]Architectures that use only information types (such as ASyMTRe) cannot distinguish between different information of the same type.

Table 7.2: Potential solutions of the navigation task

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

Table 7.3: Information used by previous approaches

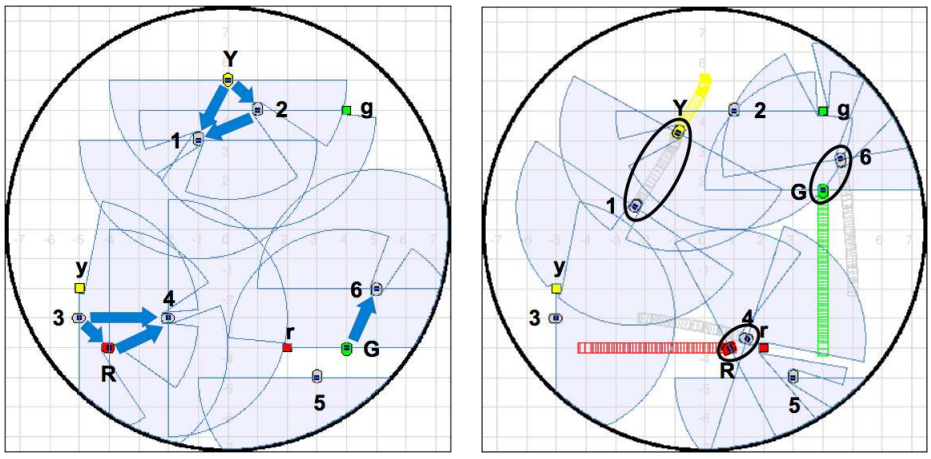
| Type | Capabilities | Count |
|---|---|---|
| 1 | Fiducial, Laser, Motor | 3 |
| 2 | Fiducial, Laser, Motor, Localization | 6 |

capability that need to navigate to certain goal positions, and there are 6 other robots that can localize. Fiducial sensors on all robots can be used to retrieve relative position information between robots. Previous approaches for forming coalitions use only information in Table 7.3, for which any type 2 robots would be considered equivalently by type 1 robots for forming coalitions, since there is no way to distinguish between them.

However, although the use of laser or localization sensors does not introduce constraints (i.e., with only 1 associated referent), a constraint on the relative position of the robots must be satisfied in order to use the fiducial sensor. Previous approaches do not consider this constraint and thus cannot form executable coalitions. IQ-ASyMTRe, on the other hand, considers it by using the available information to dynamically instantiate the potential solutions (shown in Table 7.2 in this case). For any robot $X$ that can provide $F_G(X)$, the robots also check to see if $F_R(X, local)$ or $F_R(local, X)$ can be retrieved.

For example, when the robots happen to be grouped into three clusters that are relatively distant from each other (but still within the communication range as shown in Figure 7.1(a), it is clear that the consideration of the relative positions is important. The three type 1 robots are shown in red, green and yellow in Figure 7.1(a) (labeled '$R$', '$G$', '$Y$') with the goals shown in their respective colors (labeled '$r$', '$g$', '$y$') as small square beacons. Type 2 robots are labeled from 1 to 6. The FOVs of the fiducial sensors are restricted to be 180-degree facing forward with a maximum range of 4 meters (shown as semicircles). When a robot can see another robot, an arrow is drawn between the two. Table 7.4 summarizes this scenario for coalitions, along with the overall costs for two robots. See Figures 3.5 and 7.6 for how the costs are computed[2]. Figure 7.1(b) shows the

[2]Note that how $F_A$ is retrieved is not shown in Figure 3.5. In addition to the costs of the coalitions shown, there is also a cost to activate a necessary MS (to maintain the relative position) on the robot that is chosen to help, which




(a) (b)

Figure 7.1: Forming executable coalitions in the robot navigation task. (a) IQ-ASyMTRe considers the satisfaction of the relative configurations. (b) The formed coalitions are feasible and can be executed.

Table 7.4: Coalitions with two robots in Figure 7.1

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

coalitions that are set up, indicated by ellipses. It can be concluded that IQ-ASyMTRe considers the current configurations of the robots for forming coalitions.

Note that the two potential coalitions for the red robot ('R') correspond to potential solutions 2 and 3 in Table 7.2. The coalition with a lower cost (i.e., with the red robot in the back) is preferred. For the green robot ('G'), only one executable coalition is found, although the coalition with robot 5 is close to an executable state and may be beneficial when no executable coalitions are found. In such cases, the proximity information can be used by a task planner to create an executable plan for the task. However, discussions about such scenarios are outside the scope of this research.

### The dynamic monitoring task

Next, a scenario of a monitoring task is shown in simulation involving more complicated interactions in complex environments, in which the topology of a connected sensor network[3] of robots is dynamic. In this task, 8 mobile robots are to monitor the environment while keeping a connected sensor network with fiducial sensors. When targets enter the environment (shown as black squares in Figure 7.2(a)), the robots are to provide global positions of the targets. The robots in this simulation are heterogeneous with different capabilities. The fiducial sensors on robots 1 and 2 have longer ranges with 360-degree FOVs. Only robot 4 can localize. For exploration, the robots simply use the local sensor information to search the space while maintaining a connected sensor network. Figure 7.2(b) shows a scenario of the current connected network of the robots (i.e., with network edges shown as red (narrower) arrows) and two targets ($T_1$ and $T_2$) entering the environment. In

is the same (i.e., 6) in this simulation. Finally, note that each pair of CSs connected by arrows (between the robots) in the figures is considered as one CS usage.

[3]Two robots are connected if one of them is in the other's sensor FOV.

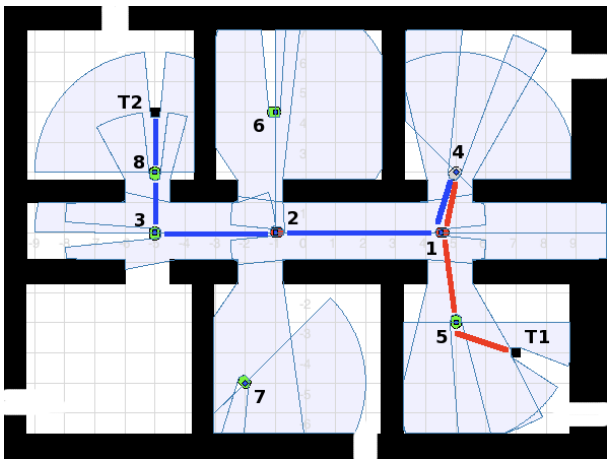
order to provide the target positions, two coalitions are dynamically formed. Members in each coalition and the respective target are connected using different line segments in Figure 7.2(a) with their interactions shown in Figure 7.2(b) as blue (wider) arrows, labeled with the communicated information instances.

One can see from this simulation that IQ-ASyMTRe can form coalitions with complicated interactions between the robots. The completeness of the solution space guarantees that a solution, when it exists, will be found given sufficient time. When the interactions must be determined dynamically based on the robot capabilities and the current configurations of the robots and the environment, it becomes impractical to always design application-specific methods; thus, general techniques such as that provided by IQ-ASyMTRe can be powerful. Another interesting aspect is that the same robots can appear in different coalitions (such as robots 1 and 4 in this simulation), such that synergy between coalitions can be achieved.

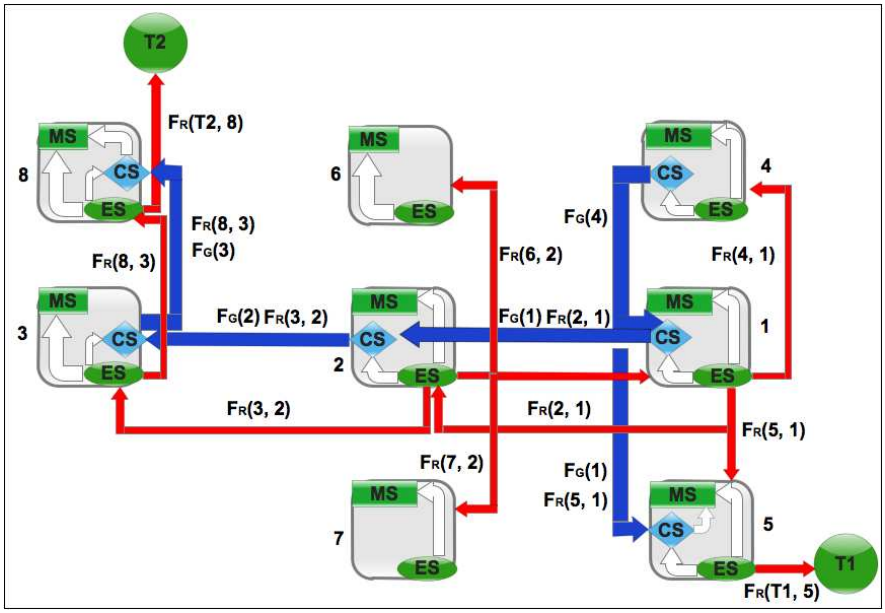
**Distributed search of the solution space**

In this simulation, an analysis on how fast IQ-ASyMTRe finds solutions distributively in difficult-to-search scenarios. A formulation of the search time is provided in general scenarios. The search process always starts with the local robot $r_L$. Denote $R_L$ as the set of robots from which $r_L$ requests help. For any robot $r_{i_1} \in R_L$, it may recursively request information from another set of robots, denoted by $R_{i_1}$; for any $r_{i_2} \in R_{i_1}$, recursively denote this set as $R_{i_1,i_2}$ and the process continues. Assuming that the maximum recursive step for the search is $N$ and that there is no message loss, the time required to find the coalition is bounded by:

$$\begin{aligned} T = & \max_{i_1:i_N} \sum_{k=1}^{N} P(r_{i_k}) \cdot S(r_{i_k}) \cdot T_G + 2 \cdot N \cdot T_C \\ & + \max_{i_1:i_N} \sum_{k=1}^{N-1} (T_D(r_{i_k}, r_{i_{k+1}}) + T_D(r_{i_{k+1}}, r_{i_k})) \end{aligned} \tag{7.1}$$




(a)




(b)

Figure 7.2: A scenario for the dynamic monitoring task with mobile robots. (a) The configuration when targets enter the environment. (b) Connections of the network and the interactions among the robots and the environment.

in which $i_1 : i_N$ are robots in a communication path, $P(r_{i_k})$ represents the index of the potential solution (after ordering) used by $r_{i_k}$ and $S(r_{i_k})$ represents the number of solution spaces created by $r_{i_k}$ during the search. $T_G$ is a constant time gap inserted between checking two consecutive potential solutions; the message queue sends received messages for processing every $T_C$ seconds. $T_D(t_1, t_2)$ is the communication delay for sending (from $t_1$) and receiving a message (by $t_2$).

To create a concrete example, a scenario is created in which all robots are aligned in a column formation and there is only one robot with a localization capability (i.e., the one in the front). All robots are assumed to be facing the front and have a fiducial to detect position information relative to nearby robots in the front. It is interesting to determine how long the robot in the back takes to find a coalition solution. Note that since the view of any robot is blocked by the one immediately in front of it, the localization capability of the front-most robot is shared in a sequential order from the nearest to the farthest robot[4].

Figure 7.3 shows the time used by the farthest robot to find the coalition solution (which must include all robots) as the number of robots increases, averaged over 5 runs. As the number of robots (i.e., equal to $N$ in Equation 7.1) increases, the number of hops that the information must travel also increases. Although the joint search space is exponential in the number of robots, one can see that the algorithm of IQ-ASyMTRe is able to distributively find the coalition involving 10 robots in about 30 seconds, when $T_G$ and $T_C$ are set to be 0.2s. In this simulation, $T_D$ is implemented to follow a normal distribution, $N(\mu, \sigma)$, in which $\mu = 3.85$ms and $\sigma = 5.05$ms. (These communication parameters are based on physical experiments with the Pioneer robots in our lab, which use the 802.11n wireless standard.) $P(r_{i_k})$ is bounded by 11 (Table 7.2) and $S(r_{i_k})$ is bounded by 10, since only global and relative position information (relative to the other 9 robots) is involved. One can see that the empirical results in Figure 7.3 are consistent with Equation 7.1.

Figure 7.3 also illustrates (in the lower two figures) the messages sent and received for the scenario with 10 robots in one of the runs. Peaks in the plots for the sent messages (the figure

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

in the center) are created when robots share the retrieved information instances with others. The farthest robot finds the coalition solution at the 28th second, which corresponds to the black peak (with circle data point) in the figure. One can also see the phase delay of the peaks while moving away from the front-most robot, which reflects the hops of information discussed earlier. Compared to the plots for the sent messages, the plots for the received messages are more in phase with each other. This is due to the fact that information requests are often addressed to all robots (whenever there are uninstantiated referents).

A 60-second gap is manually inserted to clearly separate the searching phase from the maintaining phase (after setting up a coalition). In Figure 7.3, one can also see that the numbers of sent or received messages become stable after the coalition is set up (after the 90th second), which represents the constant information flows (between different robots) required for maintaining the coalitions. One can see from this simulation that IQ-ASyMTRe can quickly provide solutions even for scenarios that are difficult to search in distributed systems.

**The cooperative robot box pushing task**

Next, how executable solutions can be used is presented to solve a cooperative robot box pushing task in a general scenario. Instead of pushing in a given direction as discussed in [Donald et al., 1997], the robots are more often required to push boxes to specified locations. For oversized boxes, it is more efficient for robots to push cooperatively, since the pushing direction of the box may need to be re-aligned frequently through rotations. The solution space for this cooperative box pushing task with two robots is presented in Figure 7.4. The bumper information is shared and used to determine whether the robot teammate is ready to push so as to synchronize their behaviors.

The initial configuration is shown in Figure 7.5(a), in which the purple box (labeled '1') needs to be pushed to the position of the blue box (labeled '2'). There are 5 robots of 3 different types in the environment. Robots from the first type (labeled 'R') are equipped with a bumper sensor for pushing the box; robots from the second type (labeled 'L') are equipped with a GPS sensor for localization; the last type (labeled 'G') has both sensors. All robots are also equipped with

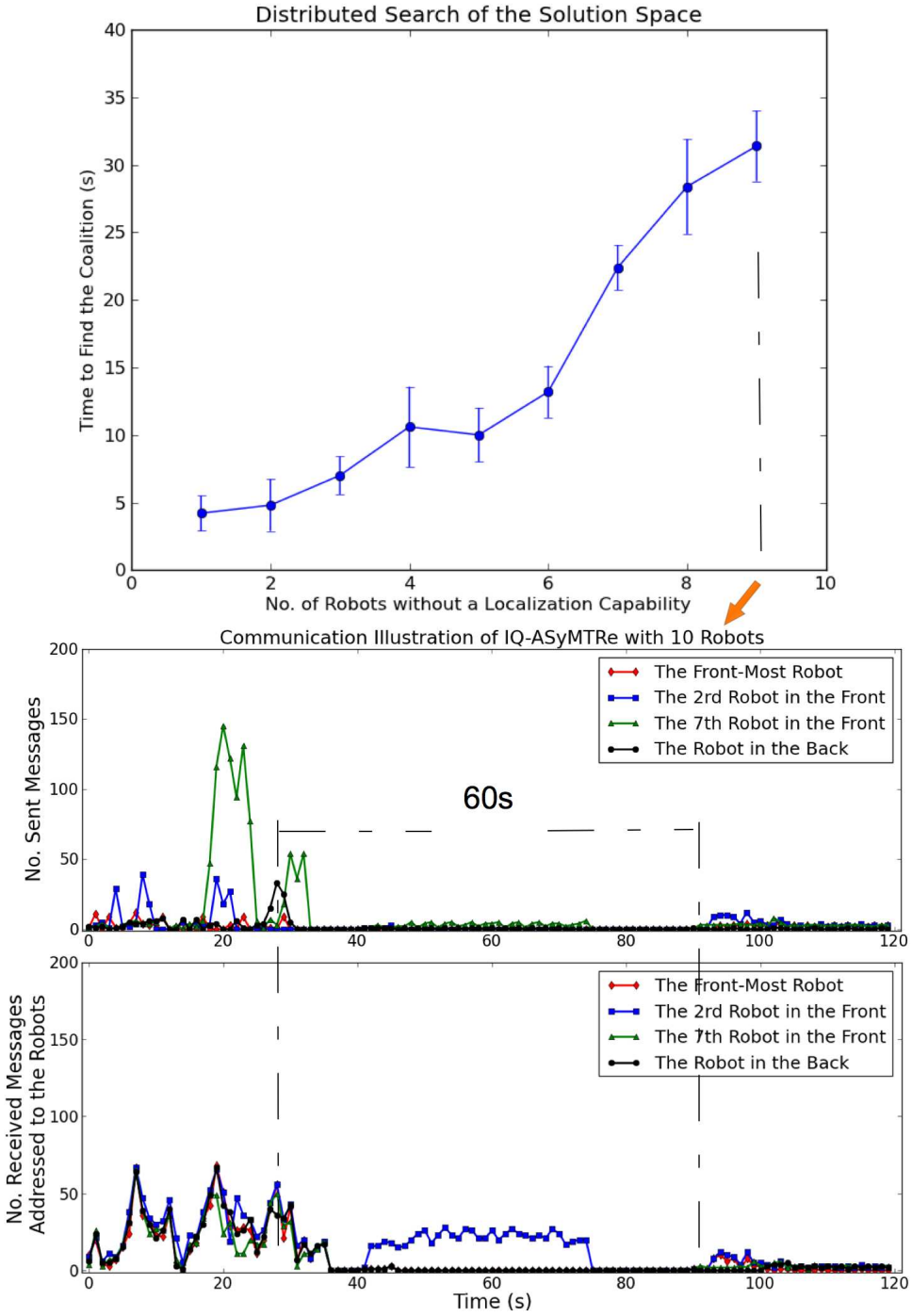



Figure 7.3: Distributed search of the solution space in difficult-to-search scenarios. The top figure shows the time to find the coalition solution as the number of robots increases. The bottom two figures show the messages sent and received (that are addressed to individual robots) for four of the robots in the 10-robot scenario.

a fiducial and a laser sensor. Since box '2' is in the way of box '1', another MS for box pushing (with a single robot) must be activated on a robot to push '2' out of the way. As a result, three robots with a bumper sensor need to be assigned (in this case, two '$R$'s and one '$G$' are assigned). However, directly executing the MSs for box pushing in the initial configuration is not possible due to the unavailability of the required information $F_G(box)$. To obtain the missing information, three MSs for navigation must be activated first on these three robots to find the boxes. Since two of the three robots cannot localize (two 'R's) , they set up a coalition with two robots that can (two 'L's), respectively. Once the boxes are found, the MSs for box pushing can be activated. Figure 7.5 shows snapshots from this simulation. A supplemental video file is attached that includes the entire execution.

To use IQ-ASyMTRe for various tasks, one only needs to implement the behaviors (MSs) and specify the required information. IQ-ASyMTRe is then able to reason about possible ways to retrieve the information in the current situation. In cases when certain information is not retrievable, higher level task planning may be required, which is manually implemented in this simulation. Creating task plans autonomously based on the missing information from IQ-ASyMTRe will be addressed in our future work.

### 7.1.2 Physical experiments

#### The cooperative robot navigation task

To demonstrate the flexibility of IQ-ASyMTRe, two scenarios are created for the navigation task with physical robots as shown in Figures 7.7 and 7.8. Instead of working with fiducial sensors, the robots are equipped with camera sensors pointing forward (with a 60-degree FOV) to retrieve the relative positions of nearby robots. The robot with a localization capability (labeled '2' ) is in front of the robot without it (labeled '1' ) in one scenario and is behind it in the other. The coalition solution for the first scenario is similar to that shown in Figure 3.5; the coalition solution for the second scenario is shown in Figure 7.6. One can compare these solutions with Figure 3.2

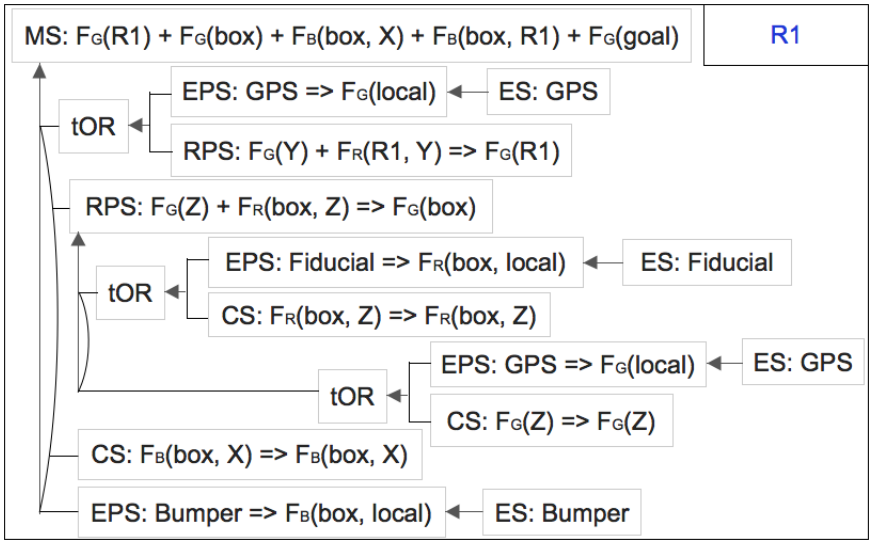
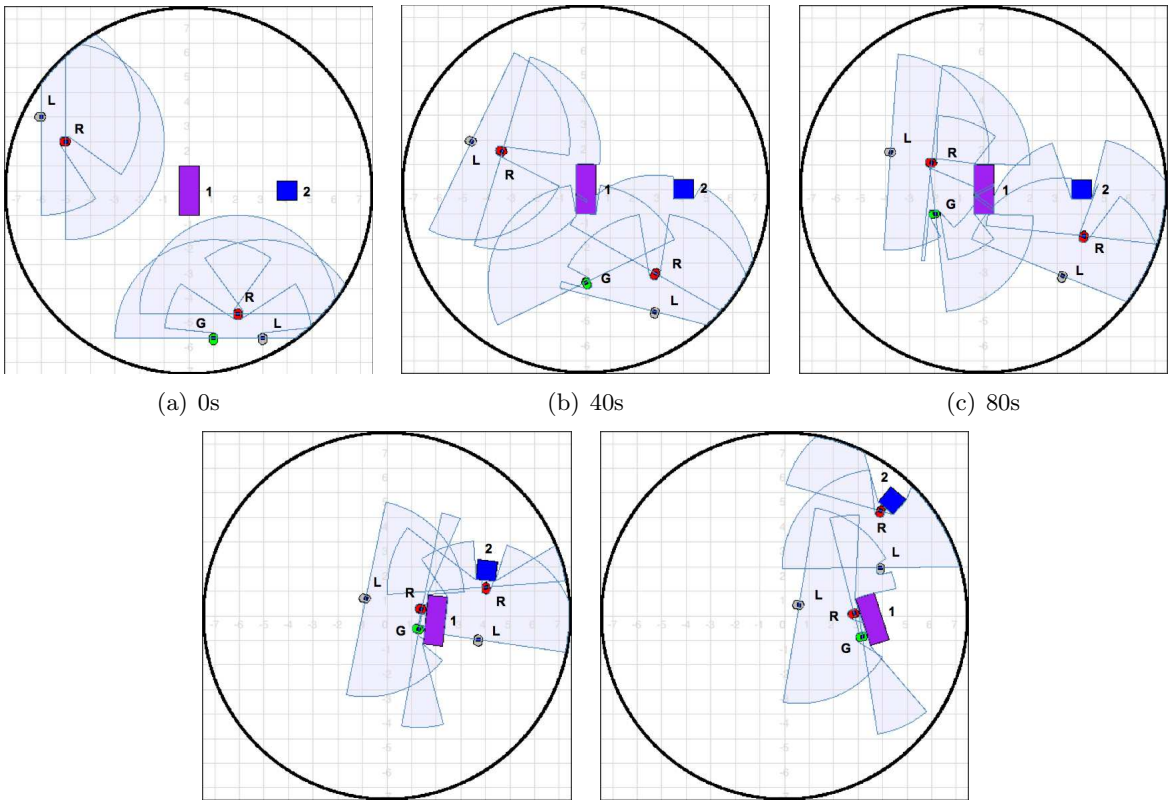



Figure 7.4: The solution space that encodes the potential solutions used in the cooperative robot box pushing task in simulation.




(a) 0s (b) 40s (c) 80s

(d) 305s (e) 635s

Figure 7.5: Robots performing a cooperative box pushing task in a general scenario, in which the goal is to push the purple box ('1') to the blue box ('2'). (a) Initial configuration in which robots are distributed in two distant clusters. Two robots with a bumper need to be assigned to activate the MS for cooperative box pushing while one is needed to push box '2' out of the way. (b) The robots ('L') with a localization capability help two of the robots assigned for box pushing to localize to find the boxes. (c) The robots are in positions ready to push. (d) The robots start pushing the boxes; two of them are being helped to achieve localization via constant information sharing; the bumper information between the two robots assigned for the cooperative box pushing is also constantly shared. (e) The task is completed.

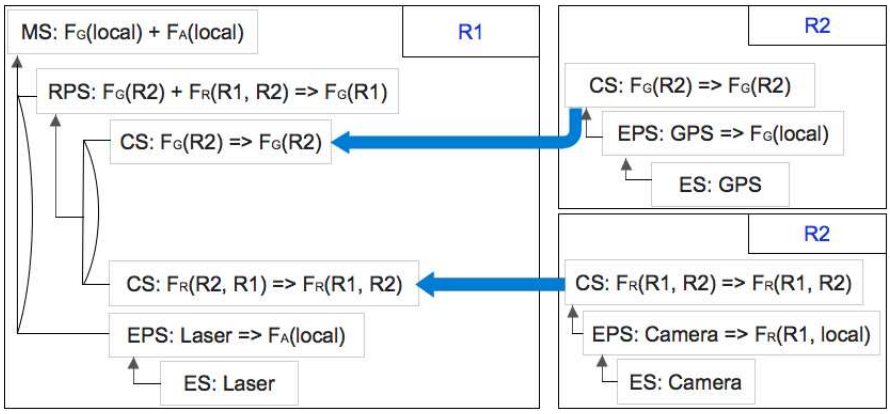



Figure 7.6: Coalition solution with the robot that can localize in the back.

to see the differences from the ASyMTRe architecture. Both robots are assigned to activate the MS for navigation to go to the same position. For both scenarios, the robot with the localization capability starts execution first since it is self-sufficient for the MS. This experiment shows that the IQ-ASyMTRe architecture can distinguish among different robot configurations in the current situations and form executable coalitions accordingly.

### The cooperative robot box pushing task

Finally, with physical robots, it is shown that IQ-ASyMTRe can provide flexible and robust solutions. Several scenarios are used to illustrate in the cooperative box pushing task with different robot capabilities and environmental settings, in which IQ-ASyMTRe provides different solutions. Unlike in the simulations, it is assumed that the robots can localize and have bumpers, so that the only information that is unavailable is $F_G(goal)$. Note that here, dynamic goal positions are assumed (e.g., pushing a box while following a person) instead of static ones. The bumper information is approximated using the sonar sensors.

Start with the simplest scenario as shown in Figure 7.9(a), in which both robots (referred to as robot $L$ and $R$) assigned to activate the MS can see the goal marker. In Figure 7.9(b), a view blocker is added to block $R$ from viewing the goal marker. Both robots are blocked from viewing


(a) (b) (c)

Figure 7.7: Robots in a navigation task, with the robot with a localization capability (labeled '2') in front. (a) Initial configurations with robot '2' in the front. (b) Robot '2' goes to the goal while the other robot without the localization capability (labeled '1') is trying to set up a coalition with it. (c) The coalition is set up and the robots navigate through the environment.


(a) (b) (c)

Figure 7.8: Robots in a navigation task, with the robot with a localization capability (labeled '2') in back. (a) Initial configurations with robot '2' at the back. (b) Robot '2' starts first while robot '1' is trying to set up a coalition with it. (c) The coalition is set up and the robots start navigation.

the goal marker in the third scenario as shown in Figure 7.9(c). Meanwhile, an intermediate robot ($I$) is added, which can localize itself and can see the goal marker. The last scenario, shown in Figure 7.9(d), is the most interesting case: the localization capability is removed from robot $I$. In this last scenario, robot $L$ obtains $F_G(goal)$ by first helping robot $I$ to localize by providing $F_R(I, L)$ and $F_G(L)$. Robot $I$ then starts helping robot $L$ in retrieving $F_G(goal)$, which, in turn, helps $R$. In this scenario, the helper also needs to be helped in order to accomplish the task. Figure 7.18 illustrates the various ways that information flows among the robots for these four scenarios.

## 7.2 Results for IQ-ASyMTRe+

Simulations and experimental results are provided to demonstrate various aspects of IQ-ASyMTRe$^+$. All reported times are wall-clock times. For simulations, the Player/Stage platform [Gerkey et al., 2003] is used; the simulations are based on a 2.4GHz Core 2 Duo laptop with 2GB memory. In these experiments, $\rho_2$ is set to 0.05, while $\rho_1$ is chosen to be a small value (0.3). The sensor quality model is implemented as $\frac{l_{max}-l}{l_{max}} \times \frac{\theta_{max}-|\theta|}{\theta_{max}}$ for both laser and camera to ensure that configurations on the sensing boundaries receive a IQ measure of 0. $M$ in the sensor uncertainty model is set to add more perturbation; for cameras, $M$ is set to be $[0.1, 0; 0, 0.1]$; for lasers, $M$ is set to be $[0.01, 0; 0, 0.01]$, since lasers are more accurate. It is noted that the sensitivity of these parameters is not high, in terms of their influences on robot behaviors.

### 7.2.1 Simulations

#### Efficient Search of the Solution Space

Let us first look at the how the approach influences the size of the solution space and how the space is searched efficiently in IQ-ASyMTRe$^+$. From Corollary 4.1.5, it holds that using the MinIIS for the MS classes can increase the number of distinct potential solutions (PoSs), hence providing more flexibility for the robots to interact with each other. Although this outcome is favorable for

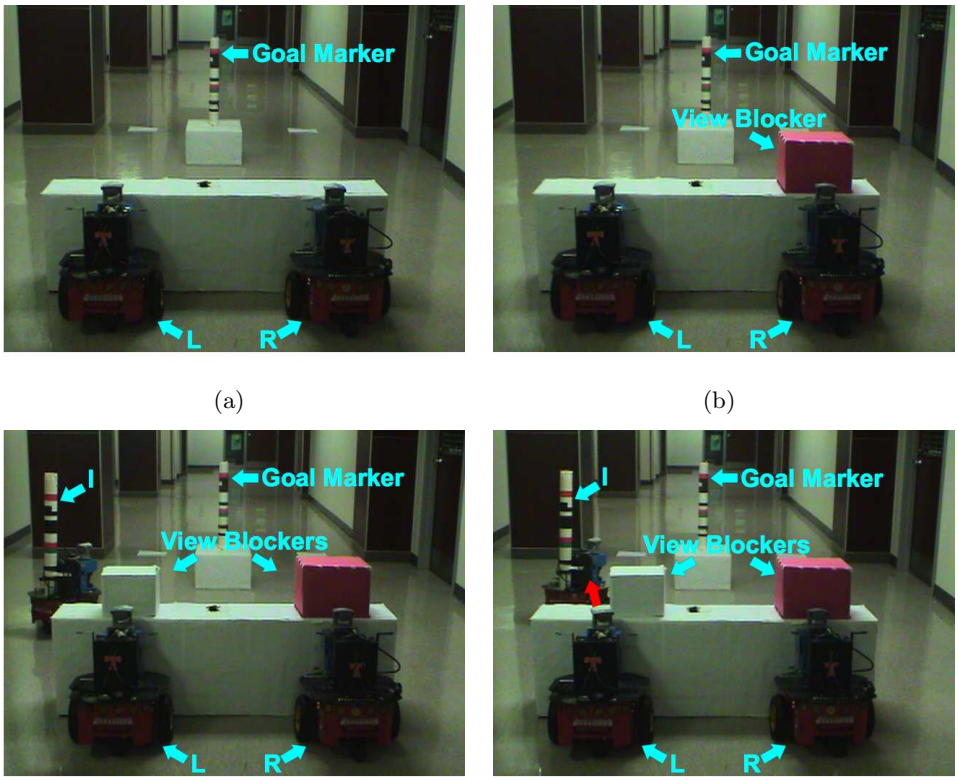



Figure 7.9: A cooperative box pushing task. (a) A simple box pushing scenario where the robots assigned to activate the MS can see the goal marker. (b) The right robot's view is blocked. (c) Both robots' views are blocked, while the intermediate robot can localize and can see the goal. (d) Both robots are blocked and the intermediate robot can see the goal but cannot localize. In all scenarios, the barcode markers are used to extract the relative position information using cameras.

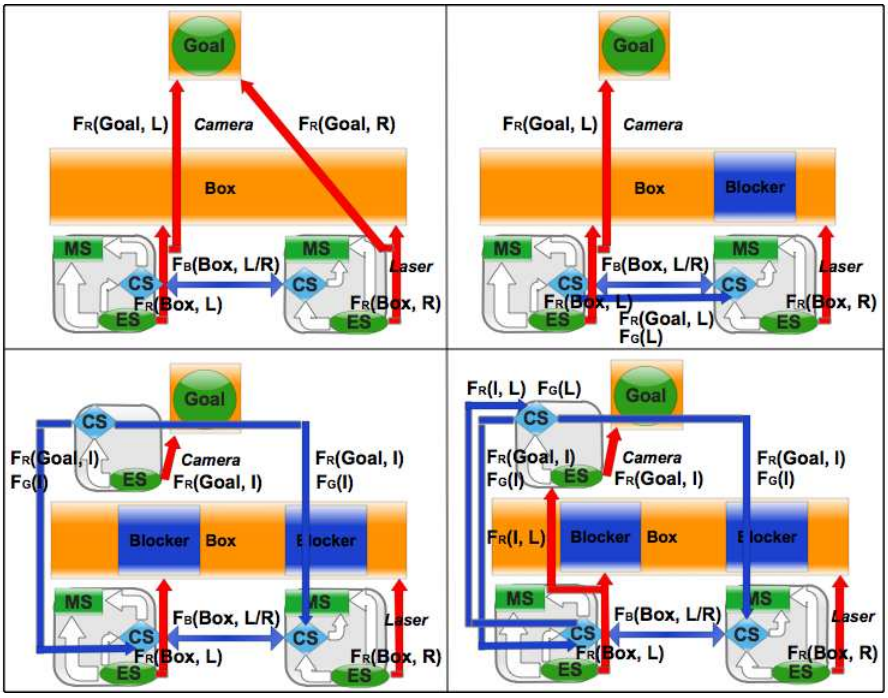



Figure 7.10: Illustration of information flow for scenarios shown in Figure 7.9.

Table 7.5: MinIIS and independence of information instance

| **goto-relative** | # PoSs | # Dist. PoSs | Use Ind. |
|---|---|---|---|
| $F_G(local)$, $F_G(goal)$ | 18 | 10 | 7 |
| $F_R(goal, local)$ | 31 | 15 | 15 |
| **push-box-to-goal** | | | |
| $F_G(local), F_G(box), F_G(goal)$ | 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 |

forming coalitions and for relaxing sensor constraints, its impact on the search performance must be analyzed. In this analysis, all RPSs in Table 3.1 are used.

This impact is analyzed for the *goto-relative* and *push-box-to-goal* MS classes and show that the independence of information instance significantly reduces the search space for IQ-ASyMTRe$^+$. For the *goto-relative* MS class, the number of PoSs is compared for one of the two MSs (i.e., requiring $\{F_G(local), F_G(goal)\}$) and for the approximated MinIIS (i.e., requiring $\{F_R(goal, local)$, $F_R(local,$ $goal)\}$), computed using Algorithm 5 for these two MSs. For the cooperative *push-box-to-goal* MS class with two robots, the most intuitive MS can use $\{F_G(local)$, $F_G(box)$, $F_G(goal)$, $F_B(box, local)$, $F_B(box, X)\}$, in which $F_B$ represents the bumper information that determines whether the robot is in contact with the box, and $X$ represents the other robot assigned the MS class. Another MS with a less stringent information requirement is $\{F_R(box, local)$, $F_R(goal, local)$, $F_B(box, local)$, $F_B(box, X)\}$, since the required IIS can be reduced from the IIS of the first MS. In this case, the approximated MinIIS computed is also the power set of the second IIS. Since $F_B$ here can only be retrieved using the bumper sensors, it is not shown in the following analysis.

Table 7.5 shows the results of the number of PoSs, the number of distinct PoSs, and the number of PoSs searched when using Definition 4.1.9 for the two MS classes. The first observation is that the results reflect the conclusion of Corollary 4.1.5. As it is demonstrated later in this section, using MinIIS provides more flexibility, which is especially useful when the capabilities of the robots are dynamic, or the environment changes. On the other hand, it is also clear that the number of PoSs

grows exponentially with the size of the IIS for the first two columns. On the other hand, when the information instances in these IISs are independent, the number of PoSs that must be searched can be significantly reduced. In such a way, IQ-ASyMTRe$^+$ achieves a much more manageable search space.

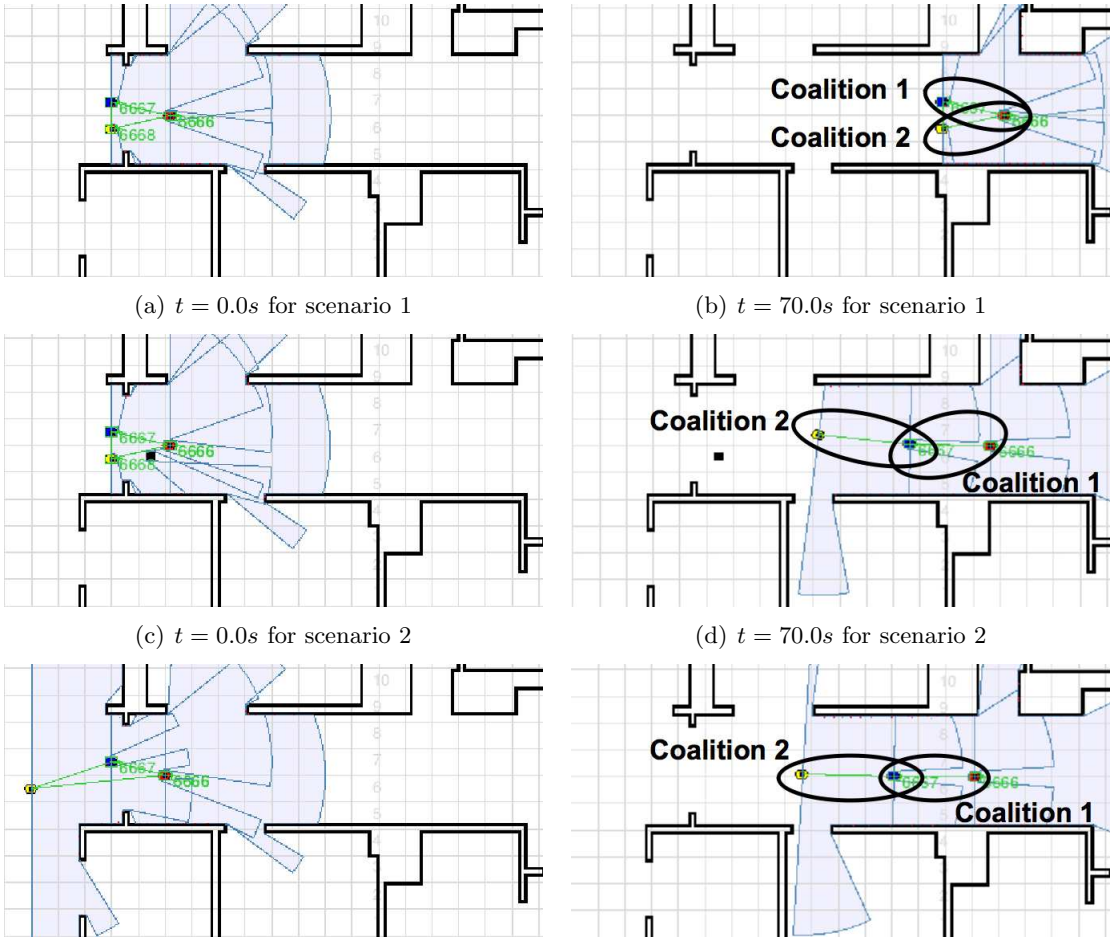
**Incorporation of the IQ Measure**

This simulation demonstrates how IQ-ASyMTRe$^+$ achieves environmental reasoning via the incorporation of the IQ measure. Using such a measure, IQ-ASyMTRe$^+$ is able to exclude coalitions of very low IQ measure from consideration. In Algorithm 7, these coalitions are identified when the IQ measure for certain required sensor constraints is below $\rho_2$. Three scenarios are created for a navigation task in Figure 7.11.

It is assumed in this simulation that three robots need to navigate to the same goal position but only one of them has a localization capability (i.e., each robot is assigned a *goto-global* MS class). The two robots without the localization capability are located to the left in Figure 7.11(a). To be able to navigate, both robots require the third robot to share its capability. In the first scenario, the robots start in a configuration as shown in Figure 7.11(a). In the second scenario (Figure 7.11(c)), an obstacle is added in the environment that partially blocks the view of one robot in the back (to the left in the figure). In the third scenario (Figure 7.11(e)), one robot in the back starts in a different position. From Figures 7.11(b) - 7.11(f), one can see that the robots are able to identify these different situations and form the desirable coalitions accordingly.

**Execution Robust to Sensor Failures**

This simulation demonstrates execution that is robust to sensor failures in a robot formation task. Robustness is achieved during execution by exploiting redundancies of robot capabilities at a more fine-grained level. The formation that is implemented is a diamond formation with four robots using a leader-reference [Balch and Arkin, 1998] strategy. One robot is assigned a *goto-global* MS class and the others are each assigned a *maintain-formation* MS class, similar to MS$^\Delta$. The simulation




(a) $t = 0.0s$ for scenario 1

(b) $t = 70.0s$ for scenario 1

(c) $t = 0.0s$ for scenario 2

(d) $t = 70.0s$ for scenario 2

(e) $t = 0.0s$ for scenario 3

(f) $t = 70.0s$ for scenario 3

Figure 7.11: Three scenarios for the robot navigation task in which robots demonstrate different behaviors. (a) Initial configuration for the first scenario. (b) Initial configuration for the second scenario, in which an obstacle is added. (c) Initial configuration for the third scenario, in which one of the robots starts in a different position. (d) Both robots in the back (to the left of the figure) set up a coalition with the robot in the front and navigate in a triangle formation. (e) The yellow robot ignores the coalition with the robot in the front due to the obstacle and sets up a coalition with the other robot in the back after the robot achieves localization. (f) The left most robot prefers the coalition with the robot in the middle and the robots navigate in a line formation.

results are presented in Figure 7.12, in which position 1 is the leader position and other robots need to maintain their relative positions. The entire execution can be divided into 3 stages as shown in the figure. The internal connections between schemas are shown as white arrows. The sensor constraints are shown as red arrows. The communication of information between robots (or CSs) are shown as blue arrows. For clarity, RPSs and EPSs are not shown. At the bottom of the figure is the number of messages sent during the execution. In the first stage, the robots form into initial coalitions; each coalition includes only a single robot. The robot at position 1 uses a GPS sensor to navigate, while the other robots use a laser fiducial sensor to retrieve the relative position information.

In the second stage (i.e., at the 75th second), the laser fiducial sensor on the robot at position 2 fails. The constraint relaxation process is dynamically triggered as the required information becomes unavailable. The constraint is relaxed at the 78th second and interactions between the robots are changed accordingly, in which the robot at position 4 communicates two pieces of relative position information to the robot at position 2. In such a way, the robot at position 2 is able to compute its own relative position to the leader. The coalition composition for this robot is expanded to include the robot at position 4. A similar process occurs at the 125th second when the laser fiducial sensor on the robot at position 3 fails. The robots again automatically reconfigure themselves. As one can see from the figure, after the new coalitions are established, the communication becomes stable, which reflects the constant information sharing between the robots.

**Execution Robust to External Influence**

The constraint relaxation process can be used similarly for achieving execution that is robust to external influence. In this simulation of the robot navigation task, three robots are to navigate to the same goal position, as shown in Figure 7.13. Among them, only the robot in the front (to the right in Figure 7.13(a)) can localize using a GPS sensor. The other two robots have a laser fiducial sensor to sense the relative positions to nearby robots. Figure 7.13 shows snapshots from the experiment with the execution time. The robots start with an initial configuration similar to

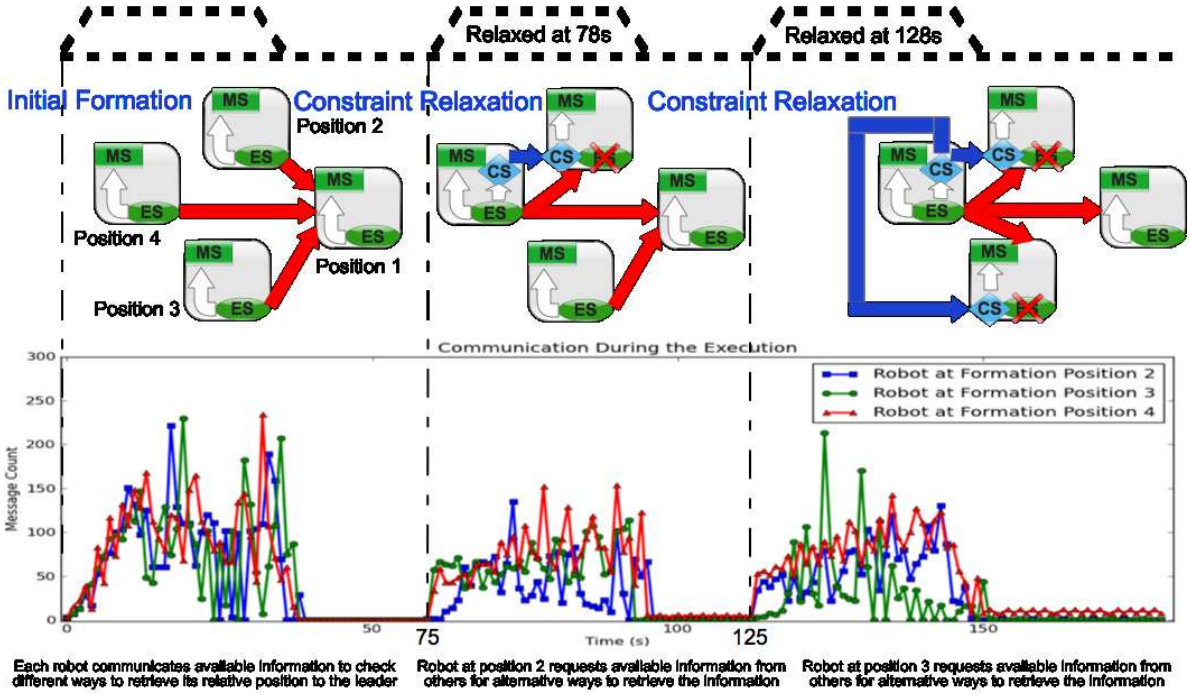


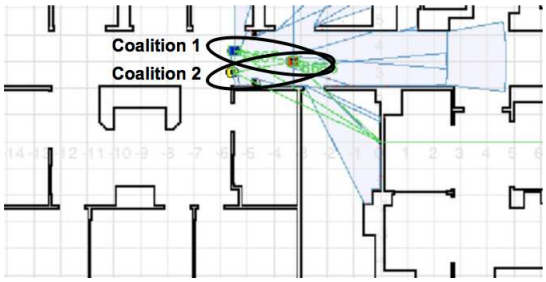

Figure 7.12: Four robots in a formation task using a leader-reference strategy. The execution can be divided into three stages. The figure shows how the interactions between the robots change as sensor failures occur during the execution.

that in Figure 7.13(a). Since both robots without the localization capability can sense the relative position to the robot in the front, they both set up a coalition with that robot. The coalitions navigate through the environment in a triangle formation as shown in Figure 7.13(a).

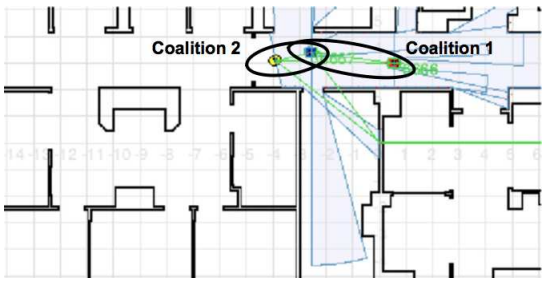
When the environment becomes narrower due to obstacles on both sides of the corridor (Figure 7.13(a)), there is risk of collision for the robot at the bottom to navigate through without adjusting its moving direction. The other following robot goes through first as the obstacles are further away. The robot at the bottom, on the other hand, first tries to improve the information quality while avoiding obstacles. However, as the quality deteriorates due to the other following robot and obstacles, the robot breaks the original coalition and triggers the constraint relaxation process. It then realizes that the other following robot can localize (with help from the robot in the front) and it is in that robot's field of view (FOV). Thus, this robot sets up a new coalition, as Figure 7.13(b) shows. Figure 7.13(c) shows that the robot coalitions navigate through the environment in a line formation and reach the goal position successfully. A supplementary video file for this entire task execution is also available for downloading.

**Flexible Execution**

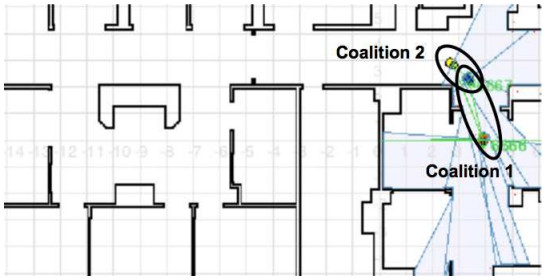
This simulation shows flexible execution is achieved in IQ-ASyMTRe$^+$ in a cooperative robot box pushing task. Unlike *push-box-to-goal*, the *push-box-in-line* MS class is implemented with two robots as in [Donald et al., 1997]. Figure 7.14 provides a solution space for a protocol in [Donald et al., 1997] to implement the MS class. The goal for this protocol is to retrieve the instantaneous torque information of the box and use the bumper sensor to determine whether the robot and the box are in contact. Since force sensing is not available, another protocol in [Donald et al., 1997] is used that is derived for quasi-static systems, in which the torque can be estimated from the position information. The IIS required by this MS is $\{F_G(box), F_G(local), F_B(box, local), F_B(box, X)\}$. The relationships between these two protocols can be summarized using the first two RPSs in Table 7.6, in which $F_I$ is the instantaneous (net) torque information.




(a) $t = 25.5s$




(b) $t = 35.5s$




(c) $t = 65.5s$

Figure 7.13: (a) Both robots without a localization capability set up a coalition with the robot in the front and robots navigate in a triangle formation. (b) The environment becomes narrower due to obstacles, influencing one of the following robots. The following robot breaks the coalition with the robot in the front and sets up a coalition with the other following robot. (c) Robots navigate through the environment in a line formation to reach the goal position.

Table 7.6: RPS's for *push-box-in-line*

| RPS |
|---|
| $F_T(Z,X) + F_T(Z,Y) \Rightarrow F_I(Z)$ |
| $F_G(X) + F_G(Y) + F_B(Z,X) + F_B(Z,Y) \Rightarrow F_I(Z)$ |
| $F_R(X,Y) + F_O(X) + F_O(Y) \Rightarrow F_B(X,Y)$ |

Applying Algorithm 5 on these two protocols with the first two RPSs in Table 7.6, the approximated MinIIS for this MS class is $\{F_I(box), F_B(box, local), F_B(box, X)\}$. The benefit is that instead of requiring either the force or position information, any method that can obtain or approximate the instantaneous torque can be used to satisfy the MinIIS, such as when an instantaneous torque sensor is installed on the box. One can see that by using MinIIS, more flexibility can be achieved for robots with varying capabilities.

A limitation of these two protocols, though, is that both require bumper sensors. To remove this reliance on particular sensors, one may use available information to approximate the sensory information through the introduction of various RPSs in IQ-ASyMTRe$^+$. For example, when the geometry ($F_O$) of the robots and the box are known, one can use the relative position information to determine whether the robots and the box are in contact by using the third RPS in Table 7.6.

The *push-box-in-line* MS class is tested with three different robot capability configurations to push the box in a given direction for 4 meters. Both robots are assigned this MS class to execute. In the first case, both robots have a bumper sensor, while only one of them has a bumper in the second case, and neither has a bumper in the third. The execution results for these three cases are presented in Figure 7.15. Note that while the task is accomplished in all cases, the cases when using the approximation are less efficient. Since approximations often require extra computation (thus incuring more costs), IQ-ASyMTRe$^+$ prefers solutions for which the required sensors are available. An interesting topic for future work is to study how RPSs influence the quality of the task execution.

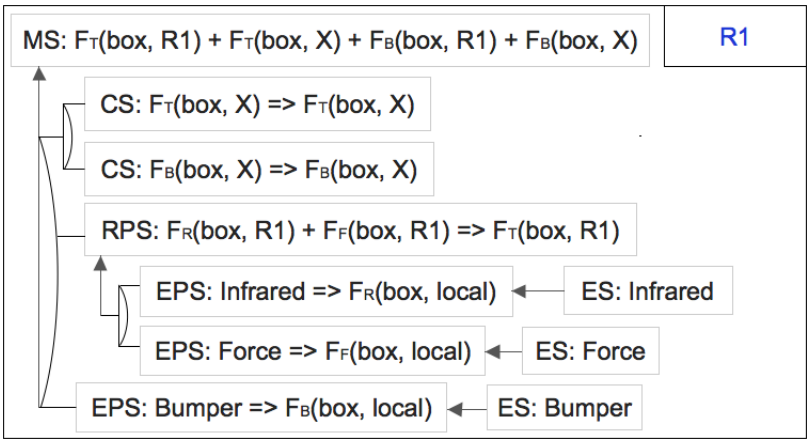



Figure 7.14: A solution space that is similar to one of the protocols presented in [Donald et al., 1997] for two robots to cooperatively push a box in a given direction. In order for the robots to coordinate actions, they estimate the net (i.e., instantaneous) torque on the box based on the torque ($F_T$) and use the bumper ($F_B$) to determine if they are in contact with the box. The torque for each robot is computed from the exerted force and the relative position to the box.

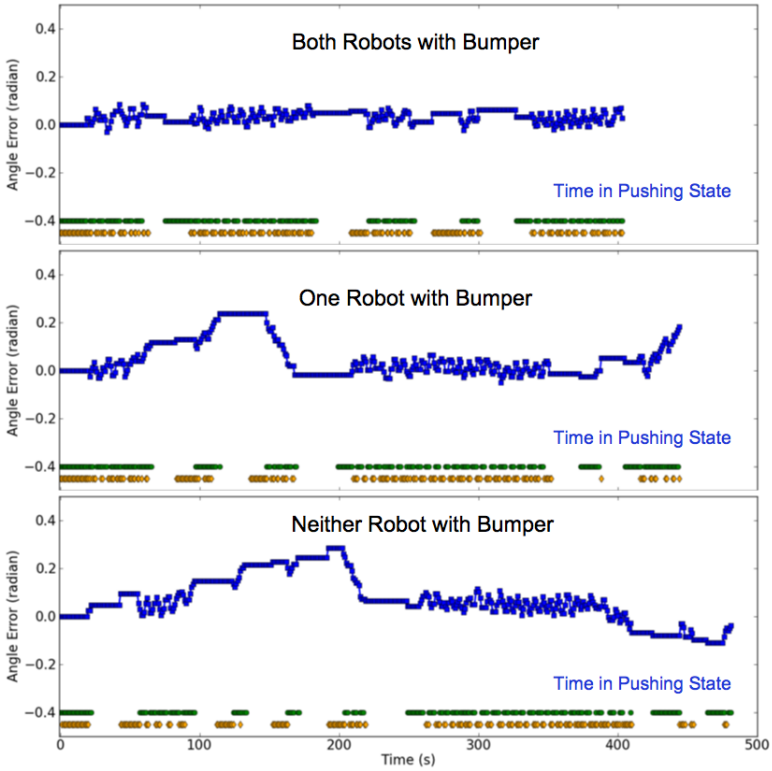



Figure 7.15: The execution of the *push-box-in-line* MS class by two robots with three different capability configurations. The goal is to push the box in a straight line for 4 meters. The angle error and distance error (perpendicular to the line of pushing) are shown for these three cases respectively.

### 7.2.2 Physical Experiments

#### Maintaining Sensor Constraints

To demonstrate IQ-ASyMTRe$^+$ for maintaining sensor constraints with physical robots, a robot tracking task is implemented. The robot simply needs to continuously execute the command vector output by Algorithm 4 to track the target, which is marked by a fiducial. For fiducial detection, the technique presented in [Parker et al., 2004] with cameras is used. This task is much more challenging with physical robots due to the FOV of the cameras, which is limited to 60 degrees in the front. To show the effectiveness of IQ-ASyMTRe$^+$, a baseline (reactive) approach is constructed that minimizes the distance and angle errors to the target while avoiding obstacles using potential fields; this baseline approach uses the vector field histogram method (VFH) [Borenstein and Koren, 1991] in Player/Stage. A similar method for considering environmental influence is used in [Das et al., 2002]. Robots in both approaches are also implemented to go to the last seen target position when the target is out of the FOV.

The robot tracking task is ran using both approaches in five different initial configurations. Table 7.7 shows the statistics for the five runs, while Figure 7.16 shows robot behaviors in one of the configurations. The blue lines represent the approximate paths taken by the tracking robots. Results that show comparable performance of IQ-ASyMTRe$^+$ with more advanced tracking approaches are presented in [Zhang and Parker, 2010a]. While the VFH approach relies on tweaking parameters (e.g., safe range) to optimize performance, IQ-ASyMTRe$^+$ uses measurable parameters for the sensors, providing a more informed solution.

One interesting note is that IQ-ASyMTRe$^+$ can be easily adapted to track multiple targets by simply averaging the IQ measures for the targets. However, when multiple tracking becomes impossible due to path divergence of the targets or environmental influence, decisions must be made in an application-specific way to choose which target to track. For example, if at least one target has to be tracked, IQ-ASyMTRe$^+$ can choose the target that is currently in the best configuration

Table 7.7: Performance comparison of IQ-ASyMTRe$^+$ and VFH approach

| Initial | IQ-ASyMTRe$^+$ | | | VFH Approach | | |
|---|---|---|---|---|---|---|
| Config. | Track Time | Time in Track | To Goal | Track Time | Time in Track | To Goal |
| Config. 1 | 30.1 | 20.2 (67%) | YES | 29.7 | 5.3 (18%) | NO |
| Config. 2 | 30.4 | 19.2 (63%) | YES | 26.5 | 9.9 (37%) | YES |
| Config. 3 | 30.0 | 17.9 (60%) | YES | 26.6 | 2.4 (9%) | NO |
| Config. 4 | 26.9 | 13.4 (50%) | YES | 18.7 | 4.1 (22%) | NO |
| Config. 5 | 27.5 | 18.8 (68%) | YES | 27.2 | 7.1 (26%) | YES |

for tracking. Another possible approach is to incorporate a learning approach as in [Parker, L.E., 2002].

**The Cooperative Box Pushing Task**

For the cooperative robot box pushing task with physical robots, both MSs for the *push-box-to-goal* MS class are implemented. Different scenarios are created and compared to reflect how the MinIIS provides more flexibility in different situations. In this experiment, the relative position from the robot to the box is retrieved using laser scan matching; the bumper information is approximated using the laser range readings. Two cases are compared in which one uses Algorithm 5 to approximate the MinIIS while the other simply selects the first MS (referred to as *Select-one*). In the first case, the robots need to consider $\{F_R(box, local), F_R(goal, local)\}$ and for the latter case, the robots need to consider $\{F_G(local), F_G(box), F_G(goal)\}$.

Let us start with the simplest scenario (shown in Figure 7.17(a)), in which the robots know where the global goal position is and can localize themselves. For the *Select-one* approach, the global positions are directly known or can be sensed, while for IQ-ASyMTRe$^+$, $F_R(goal, local)$ can be computed based on the global positions using the 4th RPS in Table 3.1.

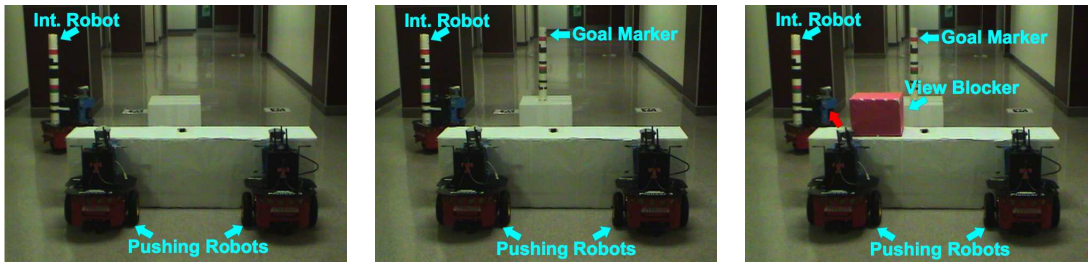
Now let us start imposing more constraints. If the global position is not known and must be observed (Figure 7.17(b)), robots using both approaches can still obtain the required information.

(a) (b) (c)

Figure 7.16: (a) An initial configuration for both approaches. (b) VFH approach: robot turns too widely when the obstacle is near; robot keeps going straight to target when the obstacle is relatively far. (c) IQ-ASyMTRe$^+$: robot keeps turning just enough to maintain tracking; robot swings aside to decrease future risk even when the obstacle is relatively far.

The global position of the goal can be computed using its relative position to the robot for the *Select-one* approach, while $F_R(goal, local)$ is directly retrievable using the camera for the IQ-ASyMTRe$^+$ approach. However, what if the robots cannot localize themselves? No solution for *Select-one* is feasible (i.e., neither $F_G(local)$ nor $F_G(local)$ can be retrieved) while IQ-ASyMTRe$^+$ is still successful.

In the next scenario (Figure 7.17(c)), the view of the goal from one of the robots is further blocked. However, the blocked robot can see an intermediate robot ($Int.$) that can detect the goal. If the robot teammate can localize itself, then both approaches can request information from $Int.$. (Note that the constraint between $Int.$ and the goal does not require a MS$^\Delta$ to be activated on $Int.$, since the goal does not move in this simulation.) However, $Int.$ may not have a localization capability. *Select-one* again would not be able to find any feasible solutions; for IQ-ASyMTRe$^+$, the blocked robot reasons out that $F_R(goal, local)$ can be obtained by using $F_R(goal, Int.)$ and $F_R(local, Int.)$ and applying the 3rd RPS in Table 3.1. In this scenario, a MS$^\Delta$ needs to be activated on $Int.$ to keep it in the sight of the pushing robot. Figure 7.18 shows the schema connections and sensor constraints for these scenarios. In this experiment, one can see the importance of choosing the proper inputs for the MSs in various situations, as it leads to the discovering of viable coalitions




(a) (b) (c)

Figure 7.17: (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.

that otherwise would not be found. More flexible execution is important for task execution with systems having different robot capabilities and in dynamic environments.

## 7.3 Simulation results for ST-MR-IA

In this section, simulation results for ST-MR-IA are provided. First, cases when the natural heuristics can produce poor quality solutions are first illustrated. Afterwards, the performance of the natural heuristics with *ResourceCentric* and *ResourceCentricApprox* in random configurations are compared. Then, simulation results with different robot capability levels for tasks are provided. Finally, results with varying maximum coalitions sizes and results with a random cost function for communication and coordination are presented. In all simulations except for the first one, or when specified otherwise, the costs of capabilities (i.e., $\boldsymbol{W}$) are randomly generated from $[0.0, 1.0]$; each robot or task has a 50% chance to have or require any capability and the capability values are randomly generated from $[0, 8]$; task rewards are randomly generated from $[100, 200]$ and $Cost$ is assumed to be a linear function of the number of robots (i.e., $4n$). All statistics are collected over 100 runs.

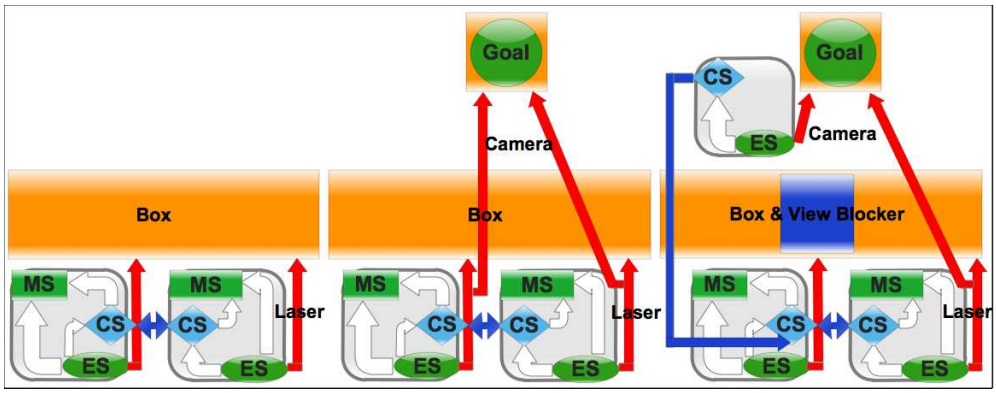
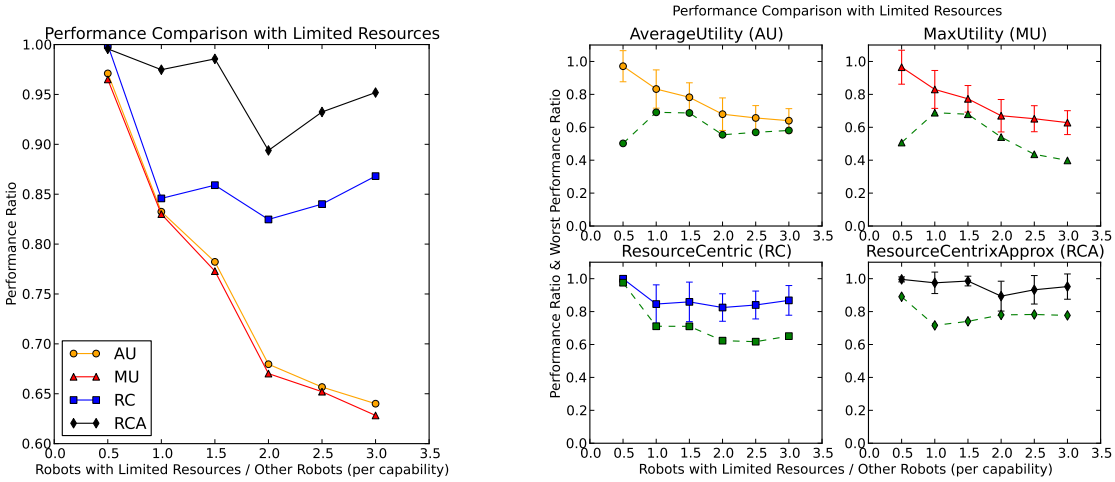



Figure 7.18: The schema connections and sensor constraints for the scenarios shown in Figure 7.17 for IQ-ASyMTRe$^+$. Illustrations from left to right correspond to Figures 7.17(a) - 7.17(c), respectively.

### 7.3.1 Comparison with limited capability resources

Based on the previous discussions, one can see that the limitation of capability resources is the key influential factor causing *AverageUtility* and *MaxUtility* to produce bad solutions. Hence in this simulation, two types of tasks are created and two limited capabilities are defined. The first type of task requires both limited capabilities and has a slightly higher reward, while the other task type requires only one of the limited capabilities. Beside the limited capabilities, two common capabilities are also defined that many robots have and most of the tasks require. It is also assumed that every robot has only one capability. For each of the two limited capabilities, only two robots are created with that capability while varying the number of robots with common capabilities. A sufficient number of tasks are generated for both types. The maximum size of the coalitions is restricted to be 3 in this simulation.

Figure 7.19 shows the results. While Figure 7.19(a) shows the average performance ratios (i.e., compared to the optimal solution), Figure 7.19(b) shows the average performance ratios with standard deviations and the worst performance ratios out of all 100 runs, separately for all four methods. One can see that as the number of robots with common capabilities increases (so that more tasks can be assigned), the ratios of *AverageUtility* and *MaxUtility* decrease drastically. This



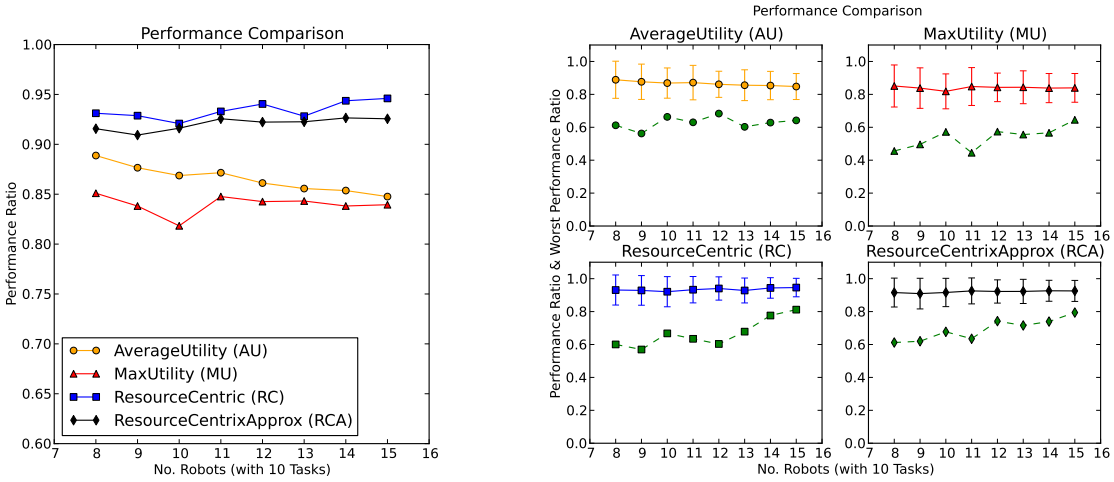

(a) (b)

Figure 7.19: Task allocation with limited capability resources. (a) Average performance ratios. (b) Separate and more detailed results with standard deviations. The green data points in each subgraph represent the worst performance ratios for that respective method.

is because both heuristics tend to choose tasks of the first type with higher rewards, although these tasks consume more of the limited capabilities such that other tasks can be significantly influenced. *ResourceCentric* and *ResourceCentricApprox*, on the other hand, consider the influence of the consumption of these limited capabilities when choosing assignments. From Figure 7.19, one can see that the performances of *AverageUtility* and *MaxUtility* keep decreasing (i.e., to around 60%) as the number of robots with common capabilities increases. Another note is that *ResourceCentricApprox* performs better than *ResourceCentric*. This is due to the fact that the measure for the greedy choice in *ResourceCentricApprox* (i.e., $\hat{\rho}$) directly optimizes on the limited capabilities (i.e., robots) in this simulation.

### 7.3.2 Comparison with random configurations

However, performances in random configurations are more interesting. In this simulation, the number of capabilities is increased to 7. In the remainder of this and the next sections, unless specified otherwise, the maximum coalition size is set to be 5. The number of robots varies while




(a) (b)

Figure 7.20: Task allocation with random configurations. (a) Average performance ratios. (b) Separate and more detailed results with standard deviations. The green data points in each subgraph represent the worst performance ratios for that respective method.

the number of tasks is fixed. Figure 7.20 shows the results. Table 7.8 shows the outcome from *t*-tests that are run to determine the statistical significance of the results in Figure 7.20. For each pair of the discussed methods, for each data point in the figure, a 'y' (yes) or 'n' (no) is used to indicate whether the results of the two methods being compared are significantly different. One can see good performances for *AverageUtility* and *MaxUtility*. However, *ResourceCentric* and *ResourceCentricApprox* still perform better. Another observation is that the new methods almost always have smaller standard deviations.

### 7.3.3 Comparison with different robot capability levels

In this simulation, results for random configurations are presented with robots of different capability levels compared to the tasks (i.e., determined by the maximum values for randomly generating the capability values). Figures 7.21(a) and 7.21(b) show the results for less and more capable robots, with maximum values of 4 and 12 respectively, while the results for the statistical significance are shown in Tables 7.9 and 7.10. Again, the number of robots varies while the number of tasks is

Table 7.8: Outcome from $t$-tests for data points (each has 100 runs) in Figure 7.20 (left to right) with $\alpha = 0.05$. Second row indicates, for each pair of methods, for each data point, whether the results are significantly different (see text for more explanation).

| **paired-sample** | $RC, AU$ | $RC, MU$ | $RCA, AU$ | $RCA, MU$ | $RC, RCA$ | $AU, MU$ |
|---|---|---|---|---|---|---|
| **sig. different?** | yyyyyyyy | yyyyyyyy | yyyyyyyy | yyyyyyyy | nnnnynyy | yyynnnnn |

Table 7.9: Outcome from $t$-tests for data points (each has 100 runs) in Figure 7.21(a) (left to right) with $\alpha = 0.05$. (See text in Section 7.3.2 for explanation of the second row.)

| **paired-sample** | $RC, AU$ | $RC, MU$ | $RCA, AU$ | $RCA, MU$ | $RC, RCA$ | $AU, MU$ |
|---|---|---|---|---|---|---|
| **sig. different?** | yyyyyyyy | yyyyyyyy | yyyyyyyy | ynnyyyyy | ynynyyyy | nnnnnnnn |

Table 7.10: Outcome from $t$-tests for data points (each has 100 runs) in Figure 7.21(b) (left to right) with $\alpha = 0.05$. (See text in Section 7.3.2 for explanation of the second row.)

| **paired-sample** | $RC, AU$ | $RC, MU$ | $RCA, AU$ | $RCA, MU$ | $RC, RCA$ | $AU, MU$ |
|---|---|---|---|---|---|---|
| **sig. different?** | yyyyyyyy | yyyyyyyy | yyyyyyyy | yyyyyyyy | nnnynyyy | nnyyyynn |

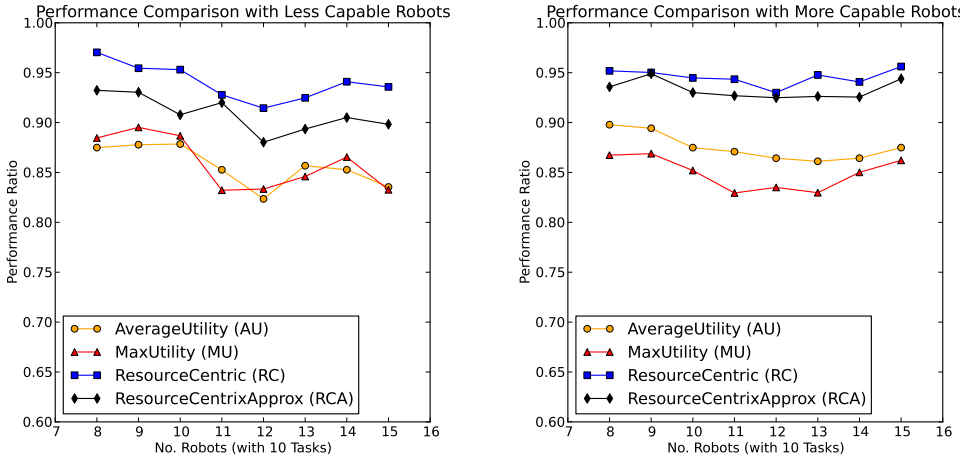
fixed. These results show that *ResourceCentric* performs the best in all cases, although not always significantly different from *ResourceCentricApprox*.

### 7.3.4 Comparison with varying coalition sizes

In this simulation, the maximum size of the coalitions varies from 3 to 11 while keeping all other settings similar to the previous simulations. Figure 7.22 and Table 7.11 show the results, which illustrate similar conclusions. While *ResourceCentric* and *ResourceCentricApprox* still perform significantly better than the other two methods, *ResourceCentric* performs only slightly better than *ResourceCentricApprox*.

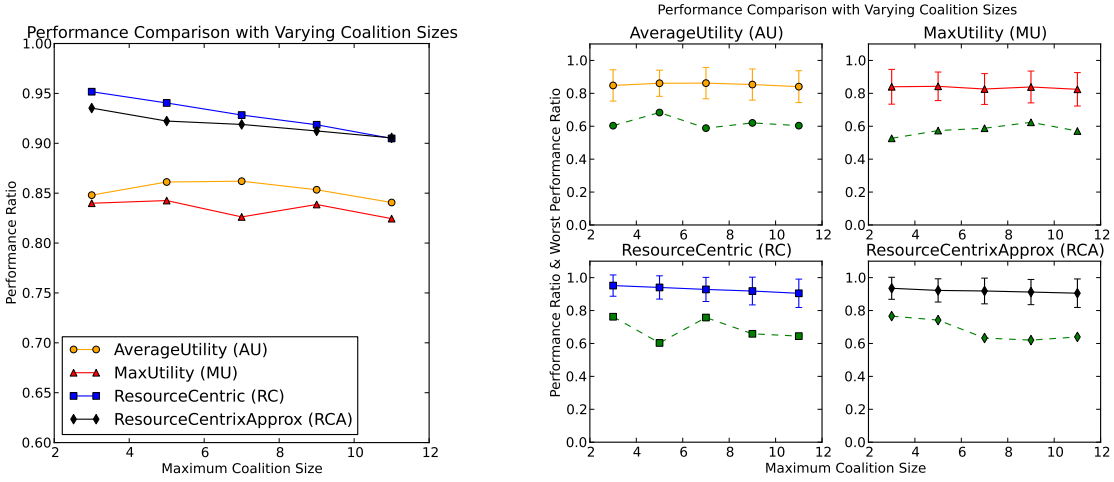
### 7.3.5 Comparison with random *Cost* function

In this simulation, the influence of the *Cost* function is investigated. Instead of defining the communication and coordination cost to be linear in the number of robots in the coalition (i.e., $4n$), *Cost* returns a random value from $[0, 4n]$. Figure 7.23 and Table 7.12 present the results.




(a) (b)

Figure 7.21: Task allocation with different robot capability levels. (a) Average performance ratios with less capable robots for tasks. (b) Average performance ratios with more capable robots for tasks.




(a) (b)

Figure 7.22: Task allocation with varying coalition sizes. (a) Average performance ratios. (b) Separate and more detailed results with standard deviations. The green data points in each subgraph represent the worst performance ratios for that respective method.

Table 7.11: Outcome from $t$-tests for data points (each has 100 runs) in Figure 7.22 (left to right) with $\alpha = 0.05$. (See text in Section 7.3.2 for explanation of the second row.)

| **paired-sample** | $RC, AU$ | $RC, MU$ | $RCA, AU$ | $RCA, MU$ | $RC, RCA$ | $AU, MU$ |
|---|---|---|---|---|---|---|
| **sig. different?** | yyyyy | yyyyy | yyyyy | yyyyy | yynnn | nnynn |

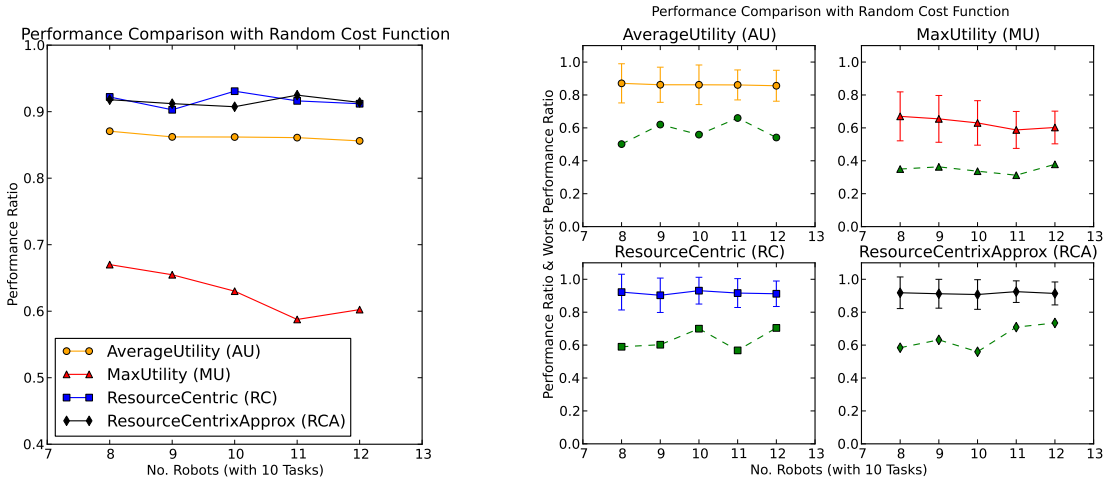
Table 7.12: Outcome from $t$-tests for data points (each has 100 runs) in Figure 7.23 (left to right) with $\alpha = 0.05$. (See text in Section 7.3.2 for explanation of the second row.)

| **paired-sample** | $RC, AU$ | $RC, MU$ | $RCA, AU$ | $RCA, MU$ | $RC, RCA$ | $AU, MU$ |
|---|---|---|---|---|---|---|
| **sig. different?** | yyyyy | yyyyy | yyyyy | yyyyy | nnynn | yyyyy |

While the conclusions regarding *ResourceCentric* and *ResourceCentricApprox* do not change, one obvious difference from the previous simulations is that the performance of *MaxUtility* significantly drops. This shows that *MaxUtility* is more sensitive to the change of the *Cost* function than the other methods. For the previous *Cost* function, a coalition with more robots is less likely to be chosen by all methods. However, when *Cost* returns a random number, *MaxUtility* cannot recognize that a coalition with fewer robots is often a better choice. For example, suppose that a task $t_1$ can be accomplished by $\{r_1, r_2\}$. As a result, $\{r_1, r_2, r_3\}$ can also accomplish the task. When the *Cost* function is linear in the number of robots in the coalitions, the coalition of $\{r_1, r_2\}$ would always be chosen by *MaxUtility*. However, when the function is random, *MaxUtility* cannot identify that $\{r_1, r_2\}$ may often be a better choice, since $r_3$ is then made available to other tasks.

### 7.3.6 Key findings from ST-MR-IA results

In this section, simulation results are provided for comparing the performances of the previously discussed methods for addressing the ST-MR-IA problem. First of all, simple scenarios are presented in which *AverageUtility* and *MaxUtility* can perform badly. Furthermore, it is shown that *ResourceCentric* and *ResourceCentricApprox*, while considering inter-task resource constraints, not only perform better in these special scenarios, but also in random configurations. This suggests




(a) (b)

Figure 7.23: Task allocation with a random cost function. (a) Average performance ratios. (b) Separate and more detailed results with standard deviations. The green data points in each subgraph represent the worst performance ratios for that respective method.

that these constraints are indeed commonly present in arbitrary configurations. Moreover, statistical testing shows that *ResourceCentric* and *ResourceCentricApprox* perform better than the other two methods with significant differences.

## 7.4 Simulation results for ST-MR-IA-TD

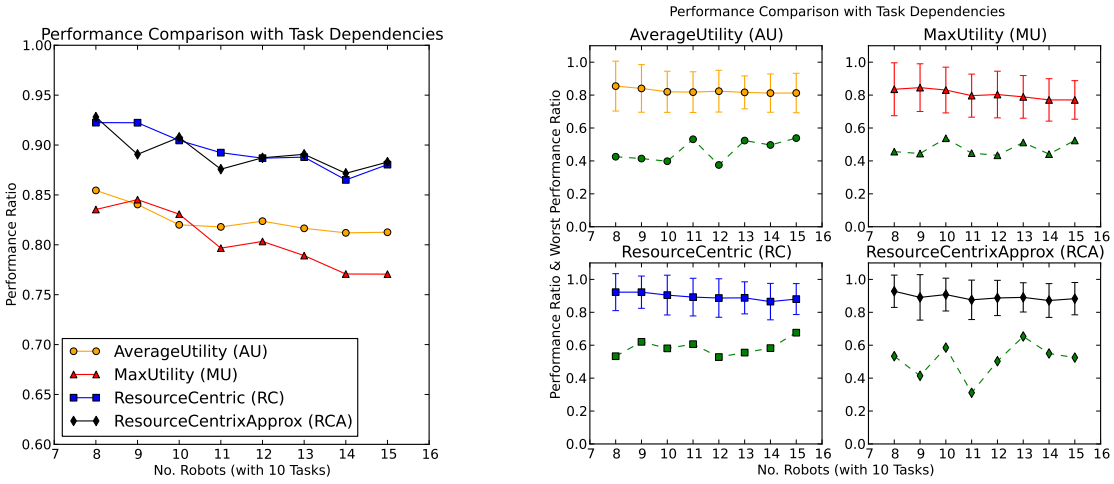
To generate dependencies for each task, in these simulations, it is assumed that the numbers of task dependencies in $\Gamma^p$ and $\Gamma^r$ are randomly chosen from $\{0, 1, 2\}$. Furthermore, for each task dependency, every other task has a probability of 0.2 to be included. Unless specified otherwise, $v_D^r$ values for tasks are randomly generated from $[0, 100]$ and $v_D^p$ values are from $[200, 400]$. After presenting simulation results with random configurations, results with varying maximum coalition sizes are shown. Results with a random cost function for communication and coordination are presented afterwards. Finally, results illustrating the influence of $v_D$ are provided and time analyses for all methods are given.

### 7.4.1 Task dependencies with random configurations

First of all, results for the ST-MR-IA-TD problem are shown with random configurations in Figure 7.24 and Table 7.13. Compared to the performance ratios of simulation results for the ST-MR-IA problem, the performance ratios are slightly worse (approximately 5% lower), which reveals that the ST-MR-IA-TD problem is indeed more difficult. Furthermore, the performances gradually decrease for all methods as the number of robots increases, such that more tasks are assigned and the influence of task dependencies becomes more prominent. Otherwise, one can still see that *ResourceCentric* and *ResourceCentricApprox* perform better than *AverageUtility* and *MaxUtility*.

### 7.4.2 Task dependencies with varying coalition sizes

Next, comparison results are shown with varying maximum coalition sizes in Figure 7.25 and Table 7.14. All methods perform similarly as for the ST-MR-IA problem (Figure 7.22), although their

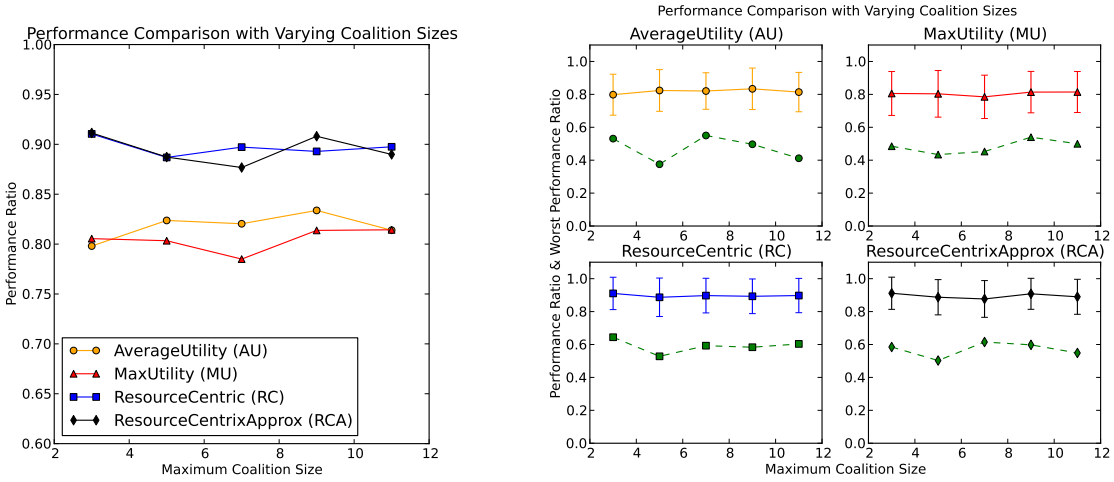



(a) (b)

Figure 7.24: Task allocation with task dependencies with random configurations. (a) Average performance ratios. (b) Separate and more detailed average performance ratios with standard deviations. The green data points in each subgraph represent the worst performance ratios for that respective method.

Table 7.13: Outcome from $t$-tests for data points (each has 100 runs) in Figure 7.24 (left to right) with $\alpha = 0.05$. (See text in Section 7.3.2 for explanation of the second row.)

| **paired-sample** | $RC, AU$ | $RC, MU$ | $RCA, AU$ | $RCA, MU$ | $RC, RCA$ | $AU, MU$ |
|---|---|---|---|---|---|---|
| **sig. different?** | yyyyyyyy | yyyyyyyy | yyyyyyyy | yyyyyyyy | nynnnnnn | nnnnnnyy |




(a) (b)

Figure 7.25: Task allocation with task dependences with varying coalition sizes. (a) Average performance ratios. (b) Separate and more detailed average performance ratios with standard deviations. The green data points in each subgraph represent the worst performance ratios for that respective method.

performances also decrease slightly for the new formulation of the problem. Again, one can see that the maximum coalition size does not influence the performance very much.

### 7.4.3 Task dependencies with random *Cost* function

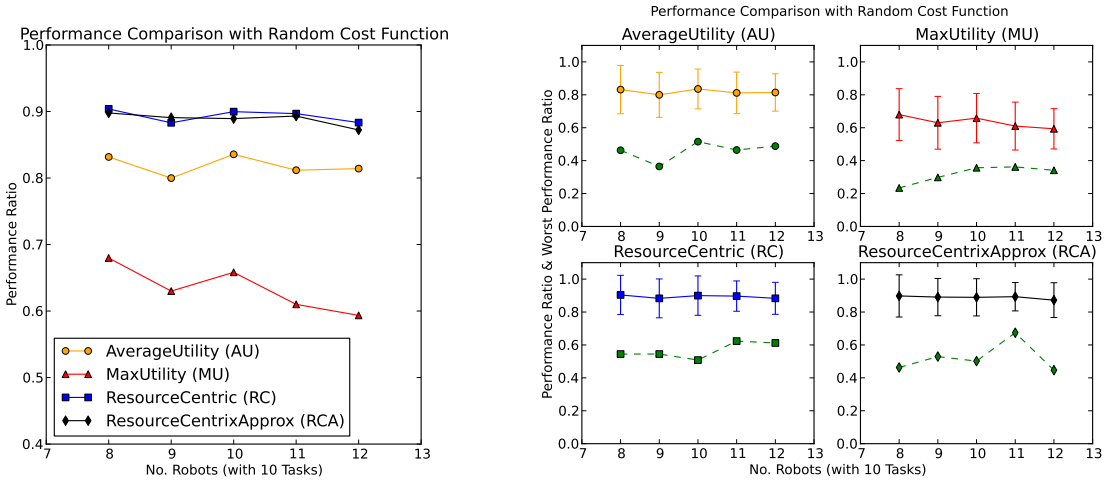
In this simulation, the *Cost* function is defined similarly as in the corresponding simulation for the ST-MR-IA problem. The results are shown in Figure 7.26 and Table 7.15. Again, one can see that *MaxUtility* is the most sensitive to the change of the *Cost* function. *ResourceCentric* and *ResourceCentricApprox* still perform better than *AverageUtility* and *MaxUtility* with significant differences, while *AverageUtility* performs better than *MaxUtility* with significant differences.

### 7.4.4 Varying maximum $v_D^p$ values of task dependencies

One can see from our previous analysis of the ST-MR-IA-TD problem that none of the methods can provide any solution guarantees that are independent of $v_D$ values. To show this effect in this

Table 7.14: Outcome from $t$-tests for data points (each has 100 runs) in Figure 7.25 (left to right) with $\alpha = 0.05$. (See text in Section 7.3.2 for explanation of the second row.)

| **paired-sample** | $RC, AU$ | $RC, MU$ | $RCA, AU$ | $RCA, MU$ | $RC, RCA$ | $AU, MU$ |
|---|---|---|---|---|---|---|
| **sig. different?** | yyyyy | yyyyy | yyyyy | yyyyy | nnnnn | nnynn |

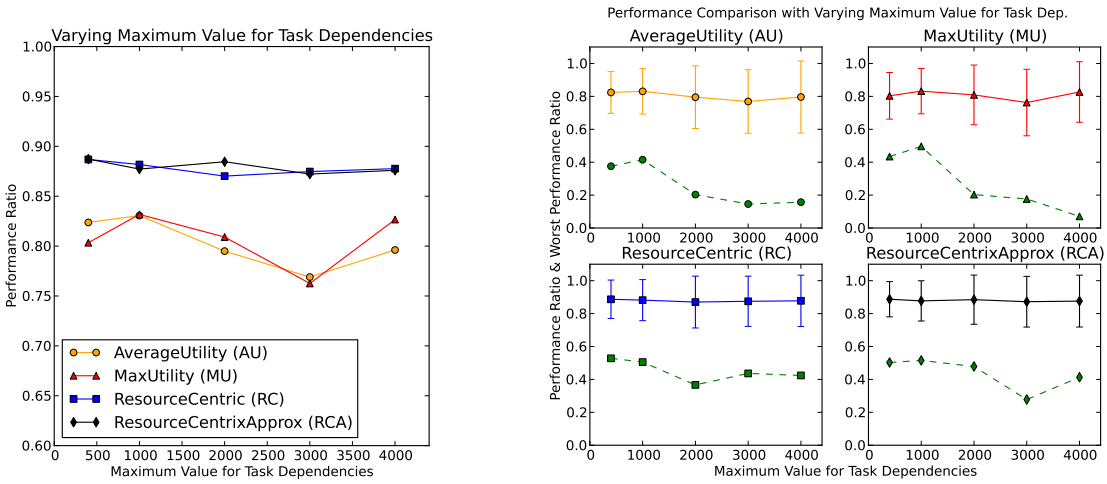



(a) (b)

Figure 7.26: Task allocation with task dependences with a random cost function. (a) Average performance ratios. (b) Separate and more detailed average performance ratios with standard deviations. The green data points in each subgraph represent the worst performance ratios for that respective method.

Table 7.15: Outcome from $t$-tests for data points (each has 100 runs) in Figure 7.26 (left to right) with $\alpha = 0.05$. (See text in Section 7.3.2 for explanation of the second row.)

| **paired-sample** | $RC, AU$ | $RC, MU$ | $RCA, AU$ | $RCA, MU$ | $RC, RCA$ | $AU, MU$ |
|---|---|---|---|---|---|---|
| **sig. different?** | yyyyy | yyyyy | yyyyy | yyyyy | nnnnn | yyyyy |




(a) (b)

Figure 7.27: Task allocation with task dependences with varying maximum value for $v_D^p$. (a) Average performance ratios. (b) Separate and more detailed average performance ratios with standard deviations. The green data points in each subgraph represent the worst performance ratios for that respective method.

simulation, the maximum value for $v_D^p$ vary in $\Gamma^p$ for tasks. For tasks without dependencies, the maximum reward value is set to be 200. Figure 7.27 and Table 7.16 show the results as the maximum value for $v_D^p$ gradually increases from 400 to 4000. While the average performance ratios remain high (with much larger standard deviations), the worst performance ratios for all four methods drop significantly as Figure 7.27(b) shows, which complies with our theoretical results. However, one can see that *ResourceCentric* and *ResourceCentricApprox* perform notably better in terms of the worst performance ratios in this simulation, especially as the maximum value for $v_D^p$ increases.

### 7.4.5 Time analysis

Finally, time analysis is provided for *AverageUtility*, *MaxUtility* and *ResourceCentricApprox* while gradually increasing the number of robots. The statistics are collected on our lab machines (2.67GHz) and the implementation is written in Java. As the time complexity of *ResourceCentric*

Table 7.16: Outcome from $t$-tests for data points (each has 100 runs) in Figure 7.27 (left to right) with $\alpha = 0.05$. (See text in Section 7.3.2 for explanation of the second row.)

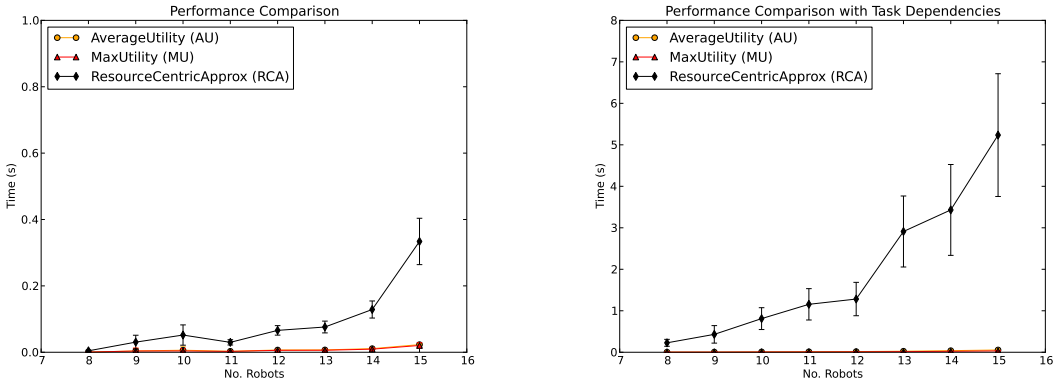
| **paired-sample** | $RC, AU$ | $RC, MU$ | $RCA, AU$ | $RCA, MU$ | $RC, RCA$ | $AU, MU$ |
|---|---|---|---|---|---|---|
| **sig. different?** | yyyyy | yyyyy | yyyyy | yyyyy | nnnnn | nnnnn |

is quadratic in $|C|$, its performance is eliminated from this first comparison for a clearer demonstration. While Figure 7.28(a) shows the results for ST-MR-IA, Figure 7.28(b) shows the results for ST-MR-IA-TD. Notice that the figures are scaled differently in this simulation. The running times of *ResourceCentricApprox* in both simulations are coarsely 100 times that of *AverageUtility* and *MaxUtility*. However, considering that $|R||T|$ is about 100 in these simulations, the results also comply with our theoretical analysis of the complexity for these methods. This suggests that *ResourceCentricApprox* indeed can be applied to problem instances of moderate sizes (e.g, 10 to 20 robots with 10 to 20 tasks to assign) within reasonable time limits (i.e., a few seconds), which is sufficient for most practical distributed robot systems.

A similar analysis is performed with *ResourceCentric* and *ResourceCentricApprox* to compare their performances. The results are shown in Figure 7.29. One can see the effect that multiplying another $|C|$ has on the time performance.

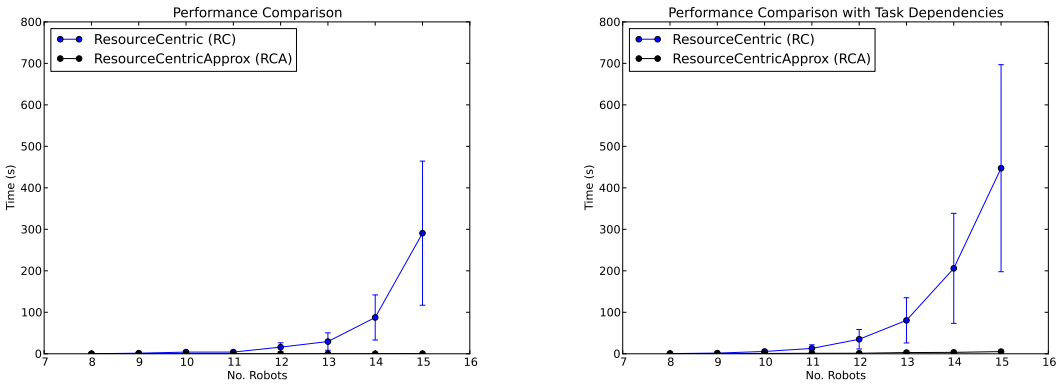
### 7.4.6 Key findings from ST-MR-IA-TD results

In this section, simulation results are provided for addressing the ST-MR-IA-TD problem. First of all, the results, along with our previous discussions, clearly demonstrate both theoretically and empirically that ST-MR-IA-TD is a more difficult problem. The result on the hardness of approximating the ST-MR-IA-TD problem is also confirmed. Furthermore, one can see that in both problem formulations, *ResourceCentric* and *ResourceCentricApprox*, which consider inter-task resource constraints, perform better than *AverageUtility* and *MaxUtility* with significant differences. Thus, when one is working with relatively small problem instances (e.g., with $< 10$ robots and $< 10$ tasks), the *ResourceCentric* heuristic is recommended, due to its solution guarantees. On the other




(a) (b)

Figure 7.28: Time analysis for all methods except *ResourceCentric*. (a) For ST-MR-IA. (b) For ST-MR-IA-TD. (Note that different scales are used for (a) and (b).)




(a) (b)

Figure 7.29: Time analysis for *RC* and *RCA*. (a) For ST-MR-IA. (b) For ST-MR-IA-TD.

hand, for problem instances of moderate sizes (i.e., with 10 to 20 robots and 10 to 20 tasks), and when the time performance is more important, *ResourceCentricApprox* is recommended.

## 7.5 Results for task allocation with executable coalitions

In this section, results for task allocation with executable coalitions and for when no executable coalitions exist are presented. In all simulations, blue robots are leader robots that have a localization capability, while red robots are follower robots. Robots in the simulations are running the same program with different configurations (e.g., communication ports) as separate processes. All data is collected based on a 2.4GHz Core 2 Duo laptop with 2GB memory and all robot processes are running on the same machine. Every robot has a laser fiducial sensor to detect other teammates. The range of these sensors is restricted to 4 meters, and the angle is restricted to 180 degrees in front of the robot.

### 7.5.1 IQ-ASyMTRe with coalition quality

First of all, different scenarios are provided in which the difficulty for finding executable coalitions gradually increases (similar to a simulation presented in Section 7.2, except that a slightly different implementation is used). As shown in Figure 7.30, the goal is for the last follower robot to achieve a localization capability. Since there is only one leader, given the configurations of the other followers, the only possible coalition is for the last follower to set up a grand coalition (in which all robots are included). Since the robots initially have only local information, the last follower must request information from others until it discovers the situation and the only coalition solution.

Figure 7.31 shows the time requirements and coalition quality measures when the number of followers is gradually increased from 1 to 9. The blue line shows the times (in seconds) that the last follower uses to find the only coalition from initially receiving the task. The coalition quality computed by the follower is also shown in red. To make the results easier to interpret, the information transferred in this simulation is not scaled down and a fixed distance for all adjacent

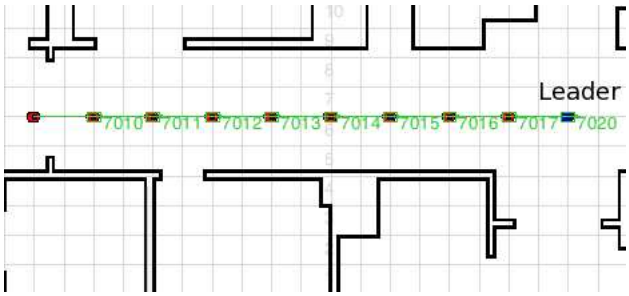



Figure 7.30: A configuration of a line of followers with one leader at the front (the right). Each robot blocks the view of the robot immediately behind it.

robots is maintained. It is also assumed that the information quality of the localization information retrieved by the leader is 1. When the last follower is the only follower (i.e., corresponding to the point with only 1 follower in Figure 7.31), the coalition quality is only influenced by the information quality of the relative position information retrieved using the fiducial sensor, which has a measure of 0.75 in this setting using the sensor models discussed in Chapter 3. When there are two followers, the coalition quality is influenced by the quality measures of both relative positions (of the same value). One can then easily induce the coalition quality with $N$ followers.

One can clearly see from Figure 7.31 that the time required to find the coalition increases as the number of followers increase. In real applications, however, the coalitions with many robots involving in such tight coordination would most likely be ignored, since the expected cost of such coalitions can easily get higher than the reward of the task (see the computation of the expected cost in Section 6.1.1). One can see from this simulation the flexibility of the IQ-ASyMTRe for finding executable coalitions and how the measure of coalition quality can help in making coalition decisions.

### 7.5.2 Executable vs. feasible coalitions

In this simulation, the advantage of task allocation with executable coalitions is demonstrated with (naturally) limited sensing capabilities (i.e., the range and angle restrictions of fiducial sensors). One observation is that robots in the multi-robot systems may often be divided into local groups

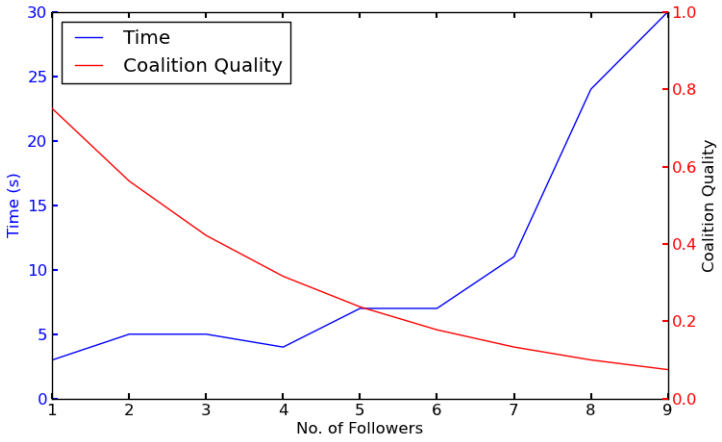



Figure 7.31: Time and coalition quality measures for the scenarios in Figure 7.30.

that are spatially separated (e.g., see Figure 7.32). Although robots in different groups may still be able to communicate, spatial separation (and other kinds of spatial restrictions) can make a large portion of the feasible coalitions not executable for the robots in many situations.

It is assumed in this simulation that there are 4 followers and 8 leaders in the environment. The task is to achieve a localization capability for all of the followers. From the given information, it is not difficult to conclude that the number of possible feasible coalitions is 3824, since any coalition that includes any leader and any follower is feasible. Furthermore, when a non-super-additive environment is assumed, the number drops to 192. This number is still not small for task allocation algorithms.

As discussed, although the number of feasible coalitions can be large, the number of executable coalitions may be limited. It is desirable to use IQ-ASyMTRe to search for executable coalitions based on the current configurations of the robots and environment. To show this, generate 10 random configurations of the robots are generated and IQ-ASyMTRe is ran to find the executable coalitions. Figure 7.33 shows 2 random configurations out of the 10 and Table 7.17 shows the results. The table shows the number of followers that can find a coalition to help it localize (i.e., **Foll. Enab.**), as well as the number of executable (**Exec.**) and feasible (**Feas.**) coalitions for any environments and non-super-additive (**n.s.a**) environments with maximum coalition size of 3.

It is obvious to see the reduction of the number of coalitions for all random configurations. One can see from this simulation that the limitation of sensing capabilities can restrict the number of coalitions that need to be considered. By using such a 'disadvantage', one can make task allocation much more efficient. One important note is that to find the executable coalitions, IQ-ASyMTRe checks only from the feasible ones. The feasibility of coalitions is automatically guaranteed by the reasoning process (by requiring necessary information to be retrieved). Furthermore, the search process for checking all feasible coalitions is naturally distributed. By trading off computation that is linear in the number of feasible coalitions, the magnitude of the possible exponential growth is reduced.

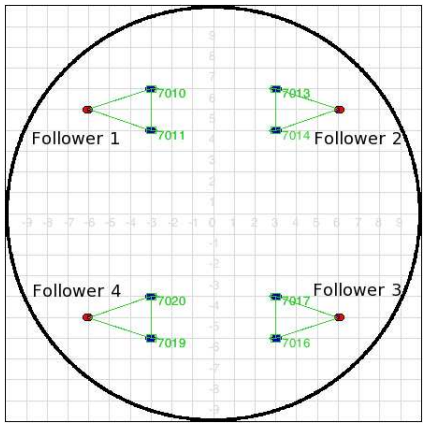
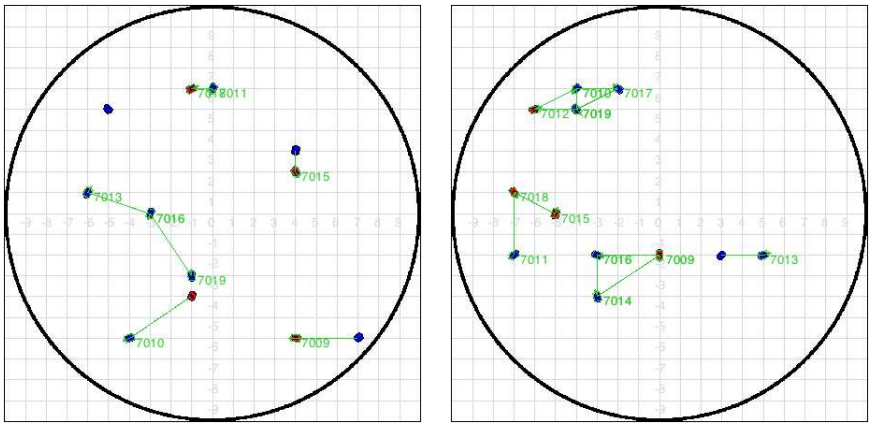



Figure 7.32: A configuration with four groups of robots spatially separated. Each group has one follower and two leaders. The range and angle restrictions of the fiducial sensors separate each one from the others.




(a) (b)

Figure 7.33: (a) A random configuration corresponding to the entry 8 in Table 7.17. (b) A random configuration corresponding to the entry 10 in Table 7.17.

Table 7.17: Executable vs. feasible coalitions

| **Conf.** | **Foll. Enab.** | **Exec.** | **Feas.** | **Exec. n.s.a** | **Feas. n.s.a** |
|---|---|---|---|---|---|
| Fig. 7.32 | 4/4 | 12 | 3824 | 12 | 192 |
| 1 | 4/4 | 33 | 3824 | 17 | 192 |
| 2 | 3/4 | 13 | 3824 | 9 | 192 |
| 3 | 2/4 | 3 | 3824 | 3 | 192 |
| 4 | 2/4 | 5 | 3824 | 3 | 192 |
| 5 | 2/4 | 6 | 3824 | 5 | 192 |
| 6 | 1/4 | 3 | 3824 | 3 | 192 |
| 7 | 2/4 | 15 | 3824 | 9 | 192 |
| 8 | 4/4 | 4 | 3824 | 4 | 192 |
| 9 | 4/4 | 11 | 3824 | 9 | 192 |
| 10 | 4/4 | 12 | 3824 | 11 | 192 |

### 7.5.3 Tasks with no executable coalitions

However, one obvious issue of task allocation with IQ-ASyMTRe is that tasks may not have executable coalitions. An approach is used that enables the robots to autonomously decompose unsatisfied preconditions of the required task behaviors into satisfiable components to create partial order plans. In this experiment, an example is provided that illustrates how such an approach works.

In this simulation (see Figure 7.34(a)), there are three tasks to be allocated and each one is for one follower to achieve a localization capability and navigate to the goal. The challenge is that one of the followers (the follower at the bottom) does not have a leader in its sight. Figure 7.34 shows the snapshots from an execution of task allocation and execution. The process can be summarized as follows:

**Figure 7.34(a)** Tasks are announced in the Easy Auction phase and the followers invoke the IQ-ASyMTRe algorithms to search for executable coalitions and submit bids for tasks. The bottom follower robot cannot find an executable coalition and hence submits no bids.

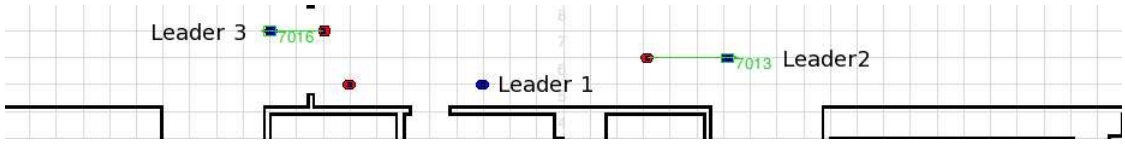
**Figure 7.34(b)** Two task assignments are made and two followers start navigating with the leaders. The auctioneer notices that no bids are submitted for a task, so it initiates the IMS Auction phase for the task. Since other followers are executing tasks, they ignore the new auction. The bottom follower receives (again) the task in the IMS Auction phase and submits information task requests to the auctioneer (i.e., *find-entity*).

**Figure 7.34(c)** The auctioneer receives the requests and announces the new task and both the bottom follower and leader submit bids. The auctioneer assigns the task to the bottom follower and it starts executing *find-entity*. (For simplicity, the potential leader is configured to be easily

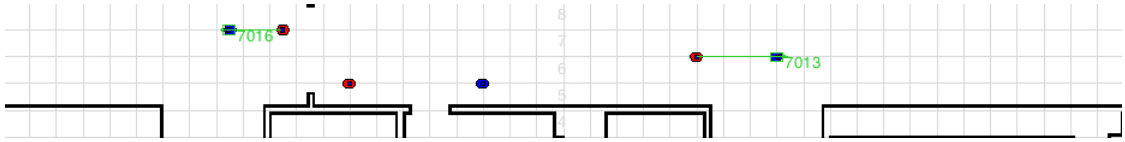
found.) Once a potential leader is found, the follower notifies the auctioneer and the auctioneer re-announces the initiating task in the Easy Auction phase, which is put on hold due to its unsatisfied precondition in the partial order plan.

**Figure 7.34(d)** The bottom follower submits a bid for the task since it now has a leader in its sight, followed by the auctioneer assigning the task to it. Finally, the bottom follower starts navigating.

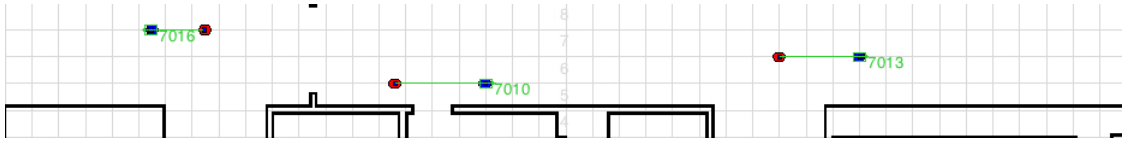
The novelty of this approach is that robots can autonomously decompose unsatisfied preconditions (i.e., input information) of the required behaviors into satisfiable components to create partial order plans, depending on the current situations. Such an approach uses the capability of IQ-ASyMTRe to reason about alternative ways to satisfy these preconditions even when no IMSs are implemented to directly satisfy them.



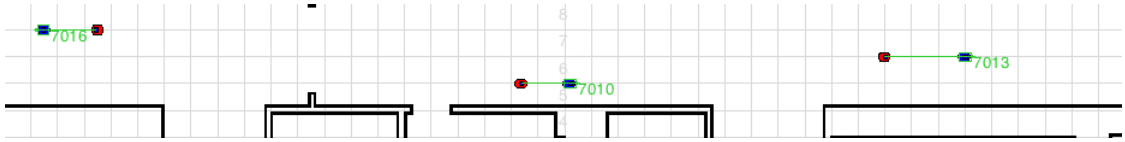

(a) 0s




(b) 20s




(c) 50s




(d) 90s

Figure 7.34: A scenario created for task allocation with 3 tasks (for the three followers to achieve a localization capability), out of which one task has no executable coalitions. The labels under each subfigure show the execution time in seconds.

# Chapter 8

# Conclusions

In this dissertation, three related problems for achieving multi-robot tasks have been discussed:

First, the IQ-ASyMTRe architecture was presented, which aims at forming executable coalitions for multi-robot tasks in which robots can also share sensory and computational capabilities. This approach significantly extends the previous ASyMTRe architecture and returns coalition solutions in which the required interactions are satisfied. The soundness and completeness of the approach were proven. Hence, IQ-ASyMTRe can be used to dynamically and flexibly form coalitions that are readily executable for different multi-robot tasks. To the best of our knowledge, this is the first attempt to create a general solution for forming executable coalitions for multi-robot tasks. Simulations and experimental results were provided to show the validity of this approach in various multi-robot tasks.

For executing coalitions, the IQ-ASyMTRe$^+$ approach for achieving flexible and robust coalition execution was presented. The IQ-ASyMTRe architecture is used to provide information of the required interactions; this information is used to coordinate the behaviors of the robots. When dynamic factors and environmental settings do not influence the execution, an established localized formation control method is used to maintain the required robot configurations; otherwise, a measure is introduced that can guide the robots to adjust their current configurations to maintain

the required interactions. A constraint relaxation process is designed to provide more robust execution when certain sensor constraints become unsatisfied, without reallocating the task. Results are provided to show the validity of our approach in various multi-robot tasks. To our knowledge, this is the first approach that provides a general method for executing coalitions in tightly-coupled multi-robot tasks.

For task allocation, an analysis of two natural heuristics for the ST-MR-IA problem was provided. A new heuristic was then presented for the problem with solution guarantees. Results show that the solution quality of this heuristic is bounded by two factors – one relates to a restricting parameter on the problem instance, while the other is influenced by how assignments in the optimal solution interact with other assignments. Note that these two factors are not bounded by each other, in the sense that while one can be greater than the other in one problem instance, it can be smaller in another. An algorithm was designed to approximate this new heuristic for performance improvement. For more complicated scenarios, the ST-MR-IA problem was extended to incorporate general task dependencies. A result on the hardness of approximating this extended formulation of the problem was given. An algorithm that utilized the methods for ST-MR-IA to address the extended problem was provided. Simulation results were presented for both formulations, which showed that these new methods indeed improved performance.

Finally, a method that combined the previous aspects for task allocation with executable coalition was presented, which serves as a starting point of a general framework to achieve system autonomy for addressing multi-robot tasks. First, how to layer the IQ-ASyMTRe architecture with task allocation was presented. Furthermore, the advantage of task allocation with executable coalitions was demonstrated in a navigation task. The reduction of the number of coalitions is the result of the limited sensing capabilities. IQ-ASyMTRe takes advantage of this 'disadvantage' and searches for executable coalitions on which task allocation is based. For tasks with no executable coalitions, a new type of MS was introduced and a process was provided that could autonomously create partial order plans to satisfy the preconditions of the required behaviors. Simulations were provided to demonstrate these techniques.

The research described in this dissertation enables autonomous task planning and execution, in which the capability of the current system is exploited for various tasks. This research represents an important step towards achieving full autonomy in distributed robot systems.

# Bibliography

# Bibliography

[Abdallah and Lesser, 2004] Abdallah, S. and Lesser, V. (2004). Organization-Based Cooperative Coalition Formation. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Techonology*, pages 162–168, China.

[Antonelli and Chiaverini, 2006] Antonelli, G. and Chiaverini, S. (2006). Kinematic control of platoons of autonomous vehicles. *IEEE Transactions on Robotics*, 22(6):1285–1292.

[Arkin, 1989] Arkin, R. C. (1989). Motor Schema – Based Mobile Robot Navigation. *International Journal of Robotics Research*, 8(4):92–112.

[Atamturk et al., 1995] Atamturk, A., Nemhauser, G. L., and Savelsbergh, M. W. P. (1995). A combined lagrangian, linear programming and implication heuristic for large-scale set partitioning problems. *Journal of Heuristics*, 1:247–259.

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

[Balch and Arkin, 1998] Balch, T. and Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939.

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

[Borenstein and Koren, 1991] Borenstein, J. and Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288.

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

[Chib and Greenberg, 1995] Chib, S. and Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335.

[Chvatal, 1979] Chvatal, V. (1979). A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3):233–235.

[Dang and Jennings, 2006] Dang, V. D. and Jennings, N. R. (2006). Coalition structure generation in task-based settings. In *Proceedings of the 17th European Conference on Artificial Intelligence*, pages 210–214.

[Das et al., 2002] Das, A. K., Fierro, R., Kumar, V., Ostrowski, J. P., Spletzer, J., and Taylor, C. J. (2002). A vision-based formation control framework. *IEEE Transactions on Robotics and Automation*, 18:813–825.

[De la cruz and Carelli, 2008] De la cruz, C. and Carelli, R. (2008). Dynamic model based formation control and obstacle avoidance of multi-robot systems. *Robotica*, 26(3):345–356.

[Desai et al., 2001] Desai, J., Ostrowski, J., and Kumar, V. (2001). Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908.

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

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

[Estlin et al., 2001] Estlin, T., Volpe, R., Nesnas, I., Mutz, D., Fisher, F., Engelhardt, B., and Chien, S. (2001). Decision-making in a robotic architecture for autonomy. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics, and Automation in Space.*

[Fanelli et al., 2006] Fanelli, L., Farinelli, A., Iocchi, L., Nardi, D., and Settembre, G. P. (2006). Ontology-based coalition formation in heterogeneous mrs. In *Proceedings of the 2006 international symposium on Practical cognitive agents and robots*, pages 105–116, New York, NY, USA. ACM.

[Fierro et al., 2001] Fierro, R., Das, A., Kumar, V., and Ostrowski, J. (2001). Hybrid control of formations of robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 157–162.

[Fikes and Nilsson, 1971] Fikes, R. E. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208.

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

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

[Garey and Johnson, 1978] Garey, M. R. and Johnson, D. S. (1978). "Strong" NP-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508.

[Gerkey and Mataric, 2000] Gerkey, B. P. and Mataric, M. J. (2000). MURDOCH: Publish/subscribe task allocation for heterogeneous agents. In *Proceedings of the 4th International Conference on Autonomous Agents*, pages 203–204. ACM Press.

[Gerkey and Mataric, 2001] Gerkey, B. P. and Mataric, M. J. (2001). Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation, Special Issue on Multi-robot Systems*.

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

[Gerkey et al., 2003] Gerkey, B. P., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323.

[Gustavi and Hu, 2005] Gustavi, T. and Hu, X. (2005). Formation control for mobile robots with limited sensor information. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1791–1796.

[Hoffman and Padberg, 1993] Hoffman, K. L. and Padberg, M. (1993). Solving airline crew scheduling problems by branch-and-cut. *Manage. Sci.*, 39:657–682.

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

[Jones et al., 2011] Jones, E., Dias, M., and Stentz, A. (2011). Time-extended multi-robot coordination for domains with intra-path constraints. *Autonomous Robots*, 30:41–56.

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

[Klusch and Gerber, 2002] Klusch, M. and Gerber, A. (2002). Dynamic coalition formation among rational agents. *IEEE Intelligent Systems*, 17.

[Lau and Zhang, 2003] Lau, H. C. and Zhang, L. (2003). Task allocation via multi-agent coalition formation: taxonomy, algorithms and complexity. In *15th IEEE International Conference on Tools with Artificial Intelligence*, pages 346–350.

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

[Lundh et al., 2007] Lundh, R., Karlsson, L., and Saffiotti, A. (2007). Plan-Based Configuration of an Ecology of Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 64–70, Rome, Italy.

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

[Michaud et al., 2002] Michaud, F., Letourneau, D., Guilbert, M., and Valin, J. (2002). Dynamic robot formations using directional visual perception. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2740–2745.

[Monteiro and Bicho, 2010] Monteiro, S. and Bicho, E. (2010). Attractor dynamics approach to formation control: theory and application. *Autonomous Robots*, 29(3-4):331–355.

[Murphy, 1998] Murphy, R. R. (1998). Dempster-Shafer theory for sensor fusion in autonomous mobile robots. *IEEE Transactions on Robotics and Automation*, 14(2):197–206.

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

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

[Parker et al., 2004] Parker, L. E., Kannan, B., Tang, F., and Bailey, M. (2004). Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 1016–1022.

[Parker et al., 2009] Parker, L. E., Reardon, C. M., Choxi, H., and Bolden, C. (2009). Using critical junctures and environmentally-dependent information for management of tightly-coupled cooperation in heterogeneous robot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2729–2736.

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

[Parker, L.E., 2002] Parker, L.E. (2002). Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12:231–255.

[Rahwan et al., 2009] Rahwan, T., Ramchurn, S. D., Jennings, N. R., and Giovannucci, A. (2009). An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, pages 521–567.

[Ren and Sorensen, 2008] Ren, W. and Sorensen, N. (2008). Distributed coordination architecture for multi-robot formation control. *Robotics and Autonomous Systems*, 56(4):324–333.

[Russell and Norvig, 2003] Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education.

[Saffiotti, 1997] Saffiotti, A. (1997). Fuzzy logic in autonomous robotics: behavior coordination. In *Proceedings of the 6th IEEE International Conference on Fuzzy Systems*, pages 573–578, Barcelona, Spain.

[Sandholm et al., 1999] Sandholm, T., Larson, K., Andersson, M., Shehory, O., and Tohme, F. (1999). Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2):209–238.

[Sandholm and Lesser, 1995] Sandholm, T. and Lesser, V. (1995). Coalition formation among bounded rational agents. *14th International Joint Conference on Artificial Intelligence*, pages 662–669.

[Sandholm et al., 2002] Sandholm, T., Suri, S., Gilpin, A., and Levine, D. (2002). Winner determination in combinatorial auction generalizations. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 69–76, New York, NY, USA. ACM.

[Sariel, 2005] Sariel, S. (2005). Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In *Integrating Planning into Scheduling: Papers from the 2005 AAAI Workshop*, pages 27–33.

[Service and Adams, 2011] Service, T. and Adams, J. (2011). Coalition formation for task allocation: theory and algorithms. *Autonomous Agents and Multi-Agent Systems*, 22:225–248.

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

[Shehory and Kraus, 1999] Shehory, O. and Kraus, T. (1999). Feasible formation of coalitions among autonomous agents in non-super-additive environments. *Computational Intelligence*, 15(1).

[Shiroma and Campos, 2009] Shiroma, P. M. and Campos, M. F. M. (2009). CoMutaR: A framework for multi-robot coordination and task allocation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4817–4824.

[Simmons, 1996] Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 3375–3382.

[Smith, 1980] Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113.

[Spletzer and Taylor, 2002] Spletzer, J. R. and Taylor, C. J. (2002). Sensor planning and control in a dynamic environment. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 676–681.

[Stroupe and Balch, 2003] Stroupe, A. and Balch, T. (2003). Value-based observation with robot teams (VBORT) using probabilistic techniques. In *Proceedings of the International Conference on Advanced Robotics*.

[Stroupe et al., 2001] Stroupe, A. W., Martin, M. C., and Balch, T. (2001). Distributed sensor fusion for object position estimation by multi-robot systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1092–1098.

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

[Tang and Parker, 2005] Tang, F. and Parker, L. E. (2005). ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1513–1520.

[Tang and Parker, 2007] Tang, F. and Parker, L. E. (2007). A complete methodology for generating multi-robot task solutions using ASyMTRe-D and market-based task allocation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3351–3358.

[Tosic and Agha, 2004] Tosic, P. T. and Agha, G. A. (2004). Maximal clique based distributed coalition formation for task allocation in large-scale multi-agent systems. In *Massively Multi-Agent Systems*, pages 104–120.

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

[Vig and Adams, 2007] Vig, L. and Adams, J. A. (2007). Coalition formation: From software agents to robots. *Journal of Intelligent and Robotic Systems*, pages 85–118.

[Werger and Mataric, 2000] Werger, B. B. and Mataric, M. J. (2000). Broadcast of local eligibility for multi-target observation. In *Proceedings of the 5th International Conference on Distributed Autonomous Robotic Systems*, pages 347–356. Springer-Verlag.

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

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

[Zhang and Parker, 2011] Zhang, Y. and Parker, L. E. (2011). Solution space reasoning to improve IQ-ASyMTRe in tightly-coupled multirobot tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[Zhang and Parker, 2012a] Zhang, Y. and Parker, L. E. (2012a). Considering inter-task resource constraints in task allocation. *Autonomous Agents and Multi-Agent Systems*.

[Zhang and Parker, 2012b] Zhang, Y. and Parker, L. E. (2012b). IQ-ASyMTRe$^+$: Achieving flexible execution for tightly-coupled multirobot tasks. *submitted and under revision for journal publication.*

[Zhang and Parker, 2012c] Zhang, Y. and Parker, L. E. (2012c). IQ-ASyMTRe: Forming executable coalitions for tightly-coupled multi-robot tasks. *submitted and under revision for journal publication.*

[Zhang and Parker, 2012d] Zhang, Y. and Parker, L. E. (2012d). Task allocation with executable coalitions in multirobot tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation.*

[Zlot and Stentz, 2005] Zlot, R. and Stentz, A. (2005). Complex task allocation for multiple robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1515–1522.

[Zlot et al., 2002] Zlot, R., Stentz, A., Dias, M. B., and Thayer, S. (2002). Multi-robot exploration controlled by a market economy. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 3016–3023.

[Zlot, 2006] Zlot, R. M. (2006). *An auction-based approach to complex task allocation for multirobot teams.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA. AAI3250901.

[Zuckerman, 2007] Zuckerman, D. (2007). Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128.

# Vita

Yu Zhang (Student Member, IEEE) received the B.S. degree in software engineering from Huazhong University of Science and Technology, Wuhan, China, in 2006 and the M.S. degree in computer science from the University of Tennessee, Knoxville (UTK), in 2009. He joined the Department of Electrical Engineering and Computer Science at UTK as a Graduate Student in August 2007. He started conducting research in the Distributed Intelligence Laboratory at UTK since 2008, working on a project funded by NSF. His research was on multi-robot cooperation and multi-agent systems. His research focuses on distributed robot systems, particularly on investigating how heterogeneous robots can reason about forming coalitions based on the current robot team configurations and environment settings, and then how robot coalitions can autonomously and flexibly execute the assigned tasks. His research interests also include multi-agent systems, planning algorithms, machine learning and sensor fusion.