**Workshop on Towards Fully Decentralized Multi-Robot Systems: Hardware, Software and Integration**
*2013 IEEE International Conference on Robotics and Automation, ICRA 2013*
*Karlsruhe, Germany, May 6 - 10, 2013,* **http://icra2013mrs.tuebingen.mpg.de/**

# A Software Suite for the Control and the Monitoring of Adaptive Robotic Ecologies (Extended Abstract)*

M. Dragone[1], M. Di Rocco[2], F. Pecora[2], A. Saffiotti[2] and D. Swords[1]

***Abstract*— Adaptive robotic ecologies are networks of heterogeneous robotic devices (sensors, actuators, automated appliances) pervasively embedded in everyday environments, where they learn to cooperate towards the achievement of complex tasks. While their flexibility makes them an increasingly popular way to improve a system's reliability, scalability, robustness and autonomy, their effective realisation demands integrated control and software solutions for the specification, integration and management of their highly heterogeneous and computational constrained components. In this extended abstract we briefly illustrate the characteristic requirements dictated by robotic ecologies, discuss our experience in developing adaptive robotic ecologies, and provide an overview of the specific solutions developed as part of the EU FP7 RUBICON Project.**

## I. INTRODUCTION

Robotic ecologies are an emerging paradigm, which crosses the border between the fields of robotics, sensor networks, and ambient intelligence (AmI). Central to the robotic ecology concept is that complex tasks are not performed by a single, very capable robot (e.g., a humanoid robot butler), instead they are performed through the collaboration and cooperation of many networked robotic devices performing several steps in a coordinated and goal oriented fashion while also exchanging sensor data and other useful information in the process. Building smart spaces in this way reduces application complexity and costs, and enhances the individual values of the devices involved, by enabling new services that cannot be performed by any device by itself. Consider for instance the case of an ecology-supported robot vacuum cleaner that avoids cleaning when any of the inhabitants are home after receiving information from the home alarm system, or of a robot informing an elderly person living alone that she has forgotten to switch off the stove, after receiving a signal from a wireless sensor installed in the kitchen. One of the key strengths of such an approach is the possibility of using alternative means to accomplish application goals when multiple courses of action are available. For instance, a robot may be able to localise itself with the help of an environmental camera or through the use of an on-board laser sensor, if the network connection to the camera is disrupted or if the light is not sufficient for the camera to track the location of the robot.

*This work was supported by the EU FP7 RUBICON project (contract n. 269914).

[1]M. Dragone and D. Swords are with the School of Computer Science and Informatics, University College Dublin, Dublin 4, Ireland mauro.dragone at ucd.ie, david.swords at ucdconnect.ie

[2]M. Di Rocco, F. Pecora and A. Saffiotti are with the Center for Applied Autonomous Sensor Systems, Örebro University, SE-70182 Sweden {modo, fpa, asaffio} at oru.se

An important prerequisite to the practical realisation of the robotic ecology concept is the necessary software infrastructure subtending the specification, integration, and the distributed management of its components. Robotic ecologies are expected to operate in dynamic, open and evolving environments; their skills are the result of many interacting components, both hardware and software. Such a multiform degree of complexity constitutes a very challenging obstacle to their effective development. Building robotic ecologies and testing them in practical experiments are costly and difficult enterprises, which require researchers to deal with a number of practical robotic and engineering issues and present them with many of the difficulties associated with open, research projects involving intelligent controllers. Using traditional robot control solutions to support the operations of robot ecologies in real home settings would quickly become ineffective, unmanageable and prohibitively costly, especially if we consider the fast innovation rates of the underlying technology, evolving settings and changing requirements, and the lack of robust control and software systems able to support the operations and the communication of information among robotic ecologies. The EU FP7 RUBICON (Robotic UBIquitous COgnitive Network) project [5] tackles these issues by endowing robotic ecologies with cognitive and learning abilities that allow them to autonomously adapt to evolving situations and achieve useful services that are not restricted to only those situations and methods that are envisioned by their designer. The aim is to greatly reduce the need of costly and long pre-programming phases and ultimately widen the levels of the user's acceptance of the technology. Instrumental to supporting those advancements, a software suite for the specification, integration and coordinated management of adaptive robotic ecologies has been developed. This has been done by building on middleware, planning and monitoring systems for robotic ecologies, and multi-agent solutions for self-adaptive and distributed software systems. The remainder of this extended abstract is organised in the following manner: Section 2 illustrates the characteristic requirements dictated by adaptive robotic ecologies. Section 3 overviews the specific solutions we have implemented and how they have been integrated into the RUBICON software suite. Section 4 briefly illustrates the work most closely related to our efforts, while Section 5 summarises our contributions and points to some of the directions to be explored in future research.

## II. REQUIREMENTS

The main goals for an adaptive robotic ecology are to provide sensing and actuating services that are *efficient*, *robust*, and *adaptable*. This requires that arbitrary combinations of a subset of the devices in the ecology should be able to be deployed in unstructured environments, such as those exemplified in a typical household, and there efficiently cooperate to the achievement of complex tasks. In doing so, robotic ecologies should take into account both a sufficient amount of exogenous events and the specific capabilities of the devices used to enact each task. To these ends, we build on learning solutions [17][6] to address the core cognitive problems of how the participants of the ecology and their capabilities should analyse the situation of the environment, decide what they need to do in order to satisfy the application's objectives, and use their past experience to drive their autonomous adaptation. However, applying a consistent approach to learning contrasts with the the many requirements that must be upheld during strategy synthesis, execution and adaptation. Specifically, robotic ecologies must be able to cope with key scientific and technical challenges in the following areas:

**Goal priorities, temporal and resource constraints** - Robot ecologies are commonly subjected to multiple, dynamically changing and possibly conflicting goals (e.g. recharge batteries versus explore, cleaning the kitchen versus asking the user if she took her medicine). To this end, they need to handle goal priorities and temporal and resource constraints, to evaluate different combinations of goals, and find the combination that will maximise the number of high-priority goals actually pursued with the resources available to the system. Since they will typically find multiple options to satisfy each goal, they need the ability to evaluate different options in terms of resource utilisation and time-frames.

**Scalability** - A robot ecology must choose a combination of hardware and software components whose exchange of information and combined ability to change the state of the environment can achieve desired goals. Robust behaviour of the ecology contrasts with the combinatorial explosion stemming from the existence of multiple options to achieve each goal. For this reason, robot ecologies needs mechanisms to promote their scalability, in terms of the number of goals that can be achieved at the same time and the number of resources that can be co-ordinated to work toward their achievement. Their operations should be robust to failures of individual devices, and their performance should degrade gracefully as the system's workload increases.

**Distribution** - At least some parts of the high-level decision making mechanisms of a robot ecology must be deployed in close proximity to its sensors and actuators in order to minimise communication and thus reduce network bandwidth usage, latency, and energy consumption, but crucially also to avoid introducing a single point of failure and subjecting each component of the ecology to the control of a centralised planner.

**Communication & re-configuration** - Robotic ecologies need to be supported by flexible communication capabilities able to connect components (both software and hardware) across different devices (robots, appliances, sensors, actuators) and allow sharing of data while changing communication path-ways to implement different tasks or in response to changing circumstances, such as when components join or leave the ecology, for instance, as a result of system maintenance, component failure, network disruptions and mobility.

**Heterogeneity** - Supporting varying computational constraints is a primary priority, as target environments will contain devices such as computers with large processing and bandwidth capacities, as well as much simpler devices such as micro-controller-based actuators and sensor nodes, and even devices with no (customisable) computational capability at all, such as Radio Frequency Identifications (RFIDs).

**Interoperability** - In order to perform useful services operating in real homes with real users, robotic ecologies should be integrated with mainstream domotic and AAL infrastructures. On one hand, the latter should be able to consider and use the capabilities of the robotic ecology, for instance, to leverage robot mobility and advanced user-system interaction, as well as their learning and reasoning mechanisms. On the other hand, robotic ecologies should be able to leverage and augment the services supported within existing smart homes, including home automation systems used to interact with embedded sensors and actuators.

**Re-use & Integration** - In order to generally reduce the complexity of their development, robot ecologies need explicit mechanisms to facilitate software re-use and integration with legacy components. Besides the obvious economic advantages associated with the re-use of existing and well-tested components (e.g. for path-planning, safe navigation, localisation and activity recognition) robotic ecologies require functional primitives to avoid the need for online learning of complex sensing-acting strategies. Rather, a robot ecology needs pre-existing modules to provide a base-line behaviour, to constrain their online exploration, but also to guarantee that the system will not behave too erratically during its initial learning stage

**Monitoring** - Notwithstanding the heterogeneity of their components, robotic ecologies need to monitor their own execution and assess their own performance in carrying out their services. This must happen not only when the robotic ecology has completed some service, or in response to failure (e.g. to trigger component and/or robot replacement), but also at runtime, for instance, in order to give interim status feedback to high-level cognitive and learning functionalities and give them the opportunity to re-assess the goals of the ecology.

## III. RUBICON CONTROL & MONITORING ARCHITECTURE AND MIDDLEWARE

In order to accommodate the above requirements, the RUBICON project has developed a software suite that includes a middleware for robotic ecologies and general planning,

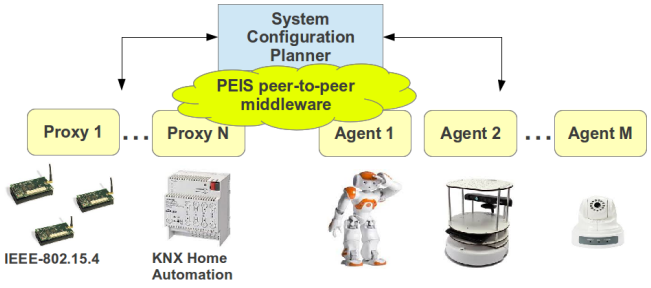


Fig. 1. Control and Monitoring System architecture

monitoring and execution modules. We employ a simple yet robust hierarchical system architecture, shown in Fig. 1, whereas a planning module oversees the main operations and the necessary collaborations among the distributed devices in the ecology - cooperating thanks to our peer-to-peer middleware. Access to heterogeneous sensing and acting networks is facilitated by a number of sensing and actuating proxy components. An agent layer, installed on each robotic device, lends each device a degree of autonomy, by instantiating and monitoring local functionalities. Such an organization is intended to simplify the job of the planner, which does not need to deal with every single device-specific detail, and to reduce the use of communication bandwidth necessary to communicate status updates between each device and the planner.

Each component in our system architecture is briefly summarised in the following section.

### A. PEIS Middleware

Interoperability among robots, wireless sensor nodes and traditional computers is ensured by the PEIS Middleware [19]. This is a fully decentralised, low footprint, portable middleware that employs a shared, distributed tuple-space as its main communication and coordination mechanism. The PEIS Middleware affords smooth integration of highly heterogeneous devices, including low-power ones like WSN Motes, spanning across multiple and heterogeneous networks; dynamic addition and removal of devices; full run-time configurability, including the replacement of devices and re-routing of messages; reflection mechanisms to inspect the available resources and their capabilities; and full decentralisation, with no broker or registry.

In order to incorporate heterogeneous and computationally constrained sensing and acting resources within a robotic ecology (including, most recently, also a range of wireless network standards and KNX home automation infrastructures), we employ a proxy design pattern, whereas a special type of PEIS component - the *proxy* component - is used to interface the robotic ecology with the sensing and acting resources available in each specific network.

Each proxy component supports the following mechanisms:

- Introspection and resource discovery mechanisms to provide an index of available sensing and acting devices, in terms of available sensor signals and accepted actuator signals. Devices in each network may join or leave the system at run-time without affecting the operations of the robotic ecology.
- Proxied sensing and actuation by accepting subscriptions to sensors automatically leading to the publication of this sensor data from the underlying sensors, and by accepting actuation signals that are transmitted to the underlying actuators.

Finally, a specific PEIS-ROS interface component is used in order to allow multiple robots equipped with ROS (Robot Operating System) to be part of a robotic ecology, i.e. to exchange data with other devices in a peer-to-peer fashion. Such a choice is dictated by the fact that ROS entails the presence of a centralized master program (ROS master) managing the connections between cooperating modules; this structure doesn't fit the purpose of having a distributed and dynamic system where robots can join and leave the network at any time.

From the point of view of the robotic ecology, each robot leveraging ROS can be seen like a monolithic structure internally constituted by a set of nodes. The data exchange between these nodes is managed by the related ROS master while the data flow within the robotic ecology is achieved through the PEIS-ROS interface (see Fig. 2).

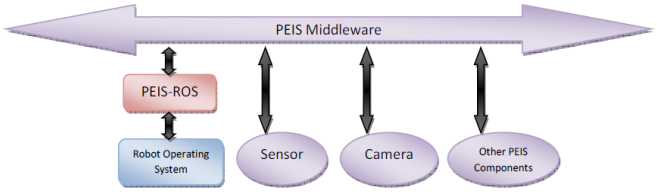


Fig. 2. the communication between ROS nodes and PEIS middleware is managed by the PEIS-ROS interface

A full description of the PEIS Middleware can be found in [7].

### B. Configuration Planner

This provides our robot ecology the ability to dynamically and autonomously re-configure to achieve the given goals using the currently available resources. The configuration planner uses the PEIS Middleware introspection mechanism to be informed about which robots and devices are available in the robot ecology, what are their capabilities, and what is their status.

A full description of the configuration planner can be found in [18]. Here we briefly summarise its most important features:

- First, our configuration planner represents activities, plans and configurations through the concept of a temporal network. This allows us to characterize activities by an explicit duration, which can be made flexible in order to absorb contingencies during the execution of the plan. It also allows us to associate temporal constraints like deadlines about task completion.

- Secondly, our system's planning and execution are tightly coupled by sharing the same temporal network. This allows the planner, among other things, to cope with multiple and dynamic goals. A goal can be posted during the execution of other ones and the planner adapts the configurations depending on the state of the execution that is continuously monitored.
- Thirdly, our framework includes provisions to explicitly reason about resource consumption: the related allocation can be modified at run time by dedicated schedulers taking into account the feedback from the sensors.
- Finally, other heterogeneous reasoners can be incorporated in addition to the schedulers, which all act on the same network (as shown in Fig. 3), which is continuously updated during the execution of the plan to be in synch with the actual environment.

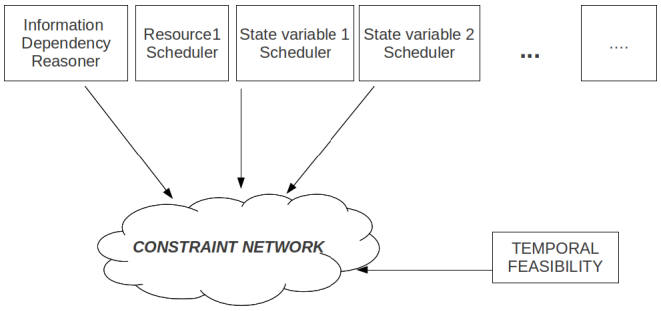


Fig. 3. Architecture of the Configuration Solver: multiple reasoners act on the same temporal network. Each modification is checked through a temporal consistency check

### C. Self-*OSGi Component & Agent Monitoring System*

The other technical challenge tackled by our software suite is the lack of a component model shared across all the heterogeneous functional components harnessed by a robotic ecology. We address this issue by building on *Self*-OSGi [9], a modular and lightweight agent & component-based framework based on the Open Service Gateway Initiative (OSGi) [2]. OSGi defines a standardised component model and a lightweight container framework, which is used as a shared platform for network-provisioned services and components specified through Java interfaces and Java classes. OSGI offers container and life cycle operations to install, start, stop and remove components together with a declarative model for automatically publishing, finding and binding their required/provided services based on XML component definitions.

OSGi specification is currently the most widely adopted technology for building modular control systems for networked home applications, with many implementations targeting computationally constrained platforms. Within the Ambient Assisted Living (AAL) domain, OSGi-based middleware have long been used to provide the technical basis for integrating network devices and services, e.g. in EU projects such as Amigo, OASIS, SOPRANO, and their recent consolidation in the UniversAAL platform.

*Self*-OSGi addresses the lack of common adaptation mechanisms in OSGi and in other component-like frameworks by injecting agent-based autonomic features to each component. Specifically, with *Self*-OSGi, each component's service requirement is treated as a goal of the agent system, which then manages the automatic and context-sensitive search for components that are able to provide each service requirement, as well as the automatic recovery from their failure. In addition, as in PEIS, *Self*-OSGi adopts a proxy design pattern to manage external components, that is, components that do not live within the OSGi container, and that communicate among each other through legacy communication mechanisms.

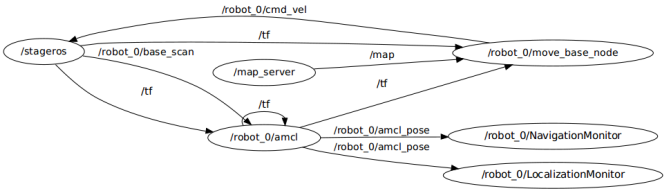


Fig. 4. Snapshot from the ROS rxgraph visualization tool showing the details of a robot navigation system dynamically assembled by the agent control system to satisfy a navigation goal request

These features are used to define an agent-based interface layer toward heterogeneous software modules, including ROS-based robotic nodes. Rather than actually implementing low-level functionalities and directly supporting their collaborations, each agent is used to manage the actual components running on the robotic devices, by re-using the communication and configuration mechanisms employed in the underlying managed system, such as topics, services and namespaces in ROS, to monitor their progress toward their objectives and their connectivity (wiring) with the other components in the system (see Fig. 4).

A full description of *Self*-OSGi and its integration with PEIS can be found in [9][8].

## IV. Related Work

Existing robot/WSN combined approaches (e.g. in related EU projects such as URUS [20], IWARD [3], GUARDIANS [1]) investigate many related issues such as cooperative monitoring, localisation and navigation. However, these issues are confronted rather in isolation, providing applied solutions that usually lack a broader applicability and that usually necessitate large scale, and costly computer systems running computationally heavy processes. Our approach in RUBICON fully embraces the robotic ecology concept, by developing robotic ecologies that exhibit tightly coupled interaction across the behaviour of all of their participants and that target computationally constrained and heterogeneous systems.

The general problem of self-configuration of a distributed system is addressed in several fields, including ambient intelligence [12][15], web service composition [11], distributed middleware and autonomic computing [10]. These works, however, do not address the same type of problem considered

here: functional coordination of a robotic ecology, in which the components of the system exchange continuous streams of data and can interact with the physical world.

The two most prominent approaches to configuration planning for multiple robots are the ASyMTRe system, and the system by Lundh and colleague. ASyMTRe [16] exploits a set of robots, each of which is equipped with a set of schemas, i.e. programs that allow the robot to: (a) acquire data from the environment, (b) process the gathered data, (c) exchange information between robots, and (d) modify the environment through actuation. A configuration represents a set of communication links that connect several schemas in order to achieve the goals. This framework has been extended to deal with multiple goals in [21] and [22], by splitting the set of robots into coalitions, which compete for tasks through an iterative auction process. The approach by Lundh and colleagues [13] considers a robot ecology deployed in a home facility. The framework leverages concepts of classical planning, and employs two interacting planners. The first is a standard action planner that builds a sequence of actions to modify the world. Such a sequence is further refined through the configuration planner that interconnects functionalities. Lundhs approach has been extended to multiple goals [14] through the use of an additional procedure: after generating stand alone configurations for each goal, a merging phase is performed to allow concurrent execution. The two above systems would not fully satisfy our configuration planning requirements, because: (1) they could not easily accommodate multiple and dynamic goals; (2) they could not represent and reason about time, which is needed in our target application scenarios; (3) they could not represent and reason about resources, which is needed to allow the concurrent execution of multiple tasks by a robot ecology; and (4) it is unclear how other forms of reasoning useful in a localized ecology, like spatial or topological reasoning, may be exploited in those frameworks.

Finally, recent efforts within the ROS community have been addressing ROS extensions to multirobot systems. For instance, the project *Rocon* [4] is an effort to make multimaster ROS as practical as possible in order to provide practical solutions to multi-robot-device-problems, e.g. involving robotic devices and mobile (tablet) user interfaces. In contrast, our design addresses highly heterogeneous robotic ecologies where there is the need to integrate multiple middleware and both robotics and mainstream (e.g. service-oriented) software offerings used in our application domains.

## V. Conclusion and Future Work

While the software described in this paper is successfully supporting our cognitive architecture, our current integrated approach has not been tested on large scale systems and still relies on a number of ad-hoc interventions to the functional components in the robotic ecology. The goal of our future work is to develop integration mechanisms that make this job as easy and automatic as possible and provide systematic evaluation of the advantages of our system architecture, for instance, in terms of scalability and robustness.

## References


[1] GUARDIANS Project. `http://www.ga-project.eu/`. [Online; Accessed, April 2013].

[2] Open Service Gateway Initiative (OSGI). `http://www.osgi.org/Main/HomePage`. [Online; Accessed, April 2013].

[3] Project IWARD. `www.iward.eu/`. [Online; Accessed, April 2013].

[4] ROCON Concert Project. `http://www.ros.org/wiki/rocon_concert`. [Online; Accessed, April 2013].

[5] G. Amato, M. Broxvall, S. Chessa, M. Dragone, C. Gennaro, R. Lopez, L. Maguire, T. M. McGinnity, A. Micheli, A. Renteria-Bilbao, and G.M.P. O'Hare. Robotic UBIquitous COgnitive Network. In *AAL Forum*, Eindhoven, NL, 24-27 September 2012.

[6] D. Bacciu, S. Chessa, C. Gallicchio, A. Lenzi, A. Micheli, and S. Pelagatti. A General Purpose Distributed Learning Model for Robotic Ecologies. In *International IFAC Symposium on Robotic Control*, volume In Robot Control 10(1), pages 435–440, 5-7 September 2012.

[7] Mathias Broxvall. The PEIS Kernel: A Middleware for Ubiquitous Robotics. In *In Proceedings of the IROS: Workshop on Ubiquitous Robotic Space Design and Applications*, San Diego, California, 29-2 October-November 2007.

[8] M. Dragone, D. Swords, S. Abdalla, M. Broxvall, and G.M.P.O'Hare. A Programming Framework for Multi Agent Coordination of Robotic Ecologies. In *AAMAS, Tenth International Workshop on Programming Multi- Agent Systems, ProMAS*, Valencia, Spain, 5 June 2012.

[9] Mauro Dragone. Component & Service-based Agent Systems: Self-OSGi. In *In Proceedings of 4th International Conference on Agents and Artificial Intelligence*, pages 200–210, Albouvera, Portugal, 6-8 February 2012.

[10] G. Tesauro et. al. A multi-agent systems approach to autonomic computing. Proc. of the Int Conf on Autonomous Agents and Multiagent Systems, pp. 464471, 2004.

[11] X. Su J. Rao. A survey of automated web service composition methods. Proc of the Int Workshop on Semantic Web Services and Web Process Composition (SWSWPC), San Diego, CA, 2004.

[12] A. Kaminsky. Infrastructure for distributed applications in ad hoc networks of small mobile wireless devices. Tech. rep., Rochester Institute of Technology, IT Lab, 2001.

[13] R. Lundh, L. Karlsson, and A. Saffiotti. Autonomous functional configuration of a network robot system. *Robotics and Autonomous Systems*, 56(10):819–830, 2008.

[14] Robert Lundh. Robert lundh. robots that help each-other: Self-configuration of distributed robot systems. In *PhD Thesis. rebro University, rebro, Sweden, May 2009.*, 2009.

[15] T. Kirste M. Hellenschmidt. Self-organization for multi-component multi-media environments. Proc of the UniComp Workshop on Ubiquitous Display Environments, 2004.

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

[17] A. K. Ray, G. Leng, T. M. McGinnity, S. Coleman, and L. Maguire. Development of Cognitive Capabilities for Smart Home using a Self-Organizing Fuzzy Neural Network. In *10th International IFAC Symposium on Robot Control*, Dubrovnik, Croatia, 5-7 September 2012.

[18] M. Di Rocco, F. Pecora, P. Kumar, and A. Saffiotti. Configuration Planning with Multiple Dynamic Goals. In *Proceedings of the AAAI Spring Symposium on Designing Intelligent Robots*, Stanford, California, March 2013.

[19] A. Saffiotti and M. Broxvall. PEIS Ecologies: Ambient intelligence meets autonomous robotics. In *In Proceedings of the International Conference on Smart Objects and Ambient Intelligence (sOc-EUSAI)*, pages 275–280, Grenoble, France, 2005.

[20] A. Sanfeliu. URUS Project (Ubiquitous Networking Robotics for Urban Settings). In *Proceedings of RISE 2008*, Benicassim (Castellón), 7-8 January 2008.

[21] Yu Zhang and L.E. Parker. Iq-asymtre: Synthesizing coalition formation and execution for tightly-coupled multirobot tasks. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5595 –5602, oct. 2010.

[22] Yu Zhang and Lynne E. Parker. Solution space reasoning to improve iq-asymtre in tightly-coupled multirobot tasks. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 370 –377, may 2011.