MULTIROBOT COMMUNICATION AND TASK COORDINATION

by

Halûk Bayram

B.S., Control and Computer Engineering, Erciyes University, 2002

M.S., Systems and Control Engineering, Boğaziçi University, 2006

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2013

MULTIROBOT COMMUNICATION AND TASK COORDINATION

APPROVED BY:

Prof. H. Işıl Bozma (Thesis Supervisor)	.....
Prof. Ayşın Ertüzün (Thesis Co-supervisor)	.....
Prof. Yağmur Denizhan	.....
Prof. Yorgo Istefanopulos	.....
Assoc. Prof. Ferit Öztürk	.....
Prof. Hakan Temeltaş	.....

DATE OF APPROVAL: 27.12.2013

ACKNOWLEDGEMENTS

I acknowledge with gratitude the invaluable guidance, support and time given in the preparation of this thesis by my advisor Prof. Işıl Bozma. I thank her for her patience and encouragement that carried me on through difficult times, and for her insights and suggestions that helped to shape my research skills. I am also thankful to my co-advisor Prof. Ayşın Ertüzün for her valuable feedbacks.

I would like to thank Prof. Yağmur Denizhan and Assoc. Prof. Ferit Öztürk for serving as a member of my thesis committee and their academic guidance. I am grateful to Prof. Yorgo Istefanopulos and Prof. Hakan Temeltaş for serving on my jury.

I would also like to extend my sincerest appreciation to my friends in Intelligent Systems Laboratory. I would like to thank Özgür Erkent and Hakan Karaoğuz for their valuable friendships and helps in this period of my life.

Last but not least, I thank my parents and family for always being there when I needed, and for supporting me.

This dissertation was supported in part by the Scientific and Technological Research Council of Turkey (107M240 and 111E285) and Bogazici University Research Fund (5169 and 5720).

ABSTRACT

MULTIROBOT COMMUNICATION AND TASK COORDINATION

This thesis is concerned with multi-robot systems. Robots are increasingly being used in applications such as patrolling or exploration whose natures necessitate the consideration of multiple cooperating robots. The cooperation may vary from sole communication to collaboration. In this perspective, the cooperation problem is examined from three critical and related aspects. First, the problem of achieving effective communication topologies is considered. Two alternative approaches - namely decentralized or centralized network topologies - are proposed based on pairwise games. Depending on the robots' objectives, the pairwise stability of the resulting networks and the convergence of the associated games are analyzed. Next, the problem of determining the assisting robots in tasks requiring the cooperation of robot pairs is considered. For this, the concept of assistance networks is introduced. Assistance networks designate potential helpmates in an implicit manner. The network topology is determined by a coordinator in a manner similar to that of centralized network topologies. Finally, the more general case of tasks requiring a multitude of resources is considered. A task coordinator is held responsible for finding robot coalitions endowed with these resources with minimal cost. This process is modeled as a series of coalition formation games. Extensive simulation and experimental results demonstrate the practical applicability of the proposed approaches in real-time multi-robot applications.

ÖZET

ÇOKLU ROBOTLARDA İLETİŞİM VE GÖREV KOORDİNASYONU

Bu tez çoklu robot sistemleri ile ilgilidir. Robotlar, devriye veya keşif gibi aralarında işbirliğini gerektiren uygulamalarda giderek artan bir şekilde kullanılmaktadır. Bu işbirliği, robotların birbirleriyle iletişiminden birlikte çalışmaya kadar uzanan farklı şekillerde olabilmektedir. Bu açıdan bakıldığında, işbirliği problemi üç kritik yönden incelenmiştir. İlk olarak, etkin iletişim topolojileri elde etme problemi ele alınmıştır. İkili oyunlara dayanan dağıtık ve merkezi olmak üzere iki yaklaşım önerilmiştir. Robotların amaçlarına bağlı olarak, oluşan ağların ikili kararlılığı ve bu oyunların yakınsaması incelenmiştir. Sonrasında, robot çiftlerinin işbirliğini gerektiren görevlerde yardımcı robotların belirlenmesi problemi ele alınarak, yardım ağları kavramı sunulmuştur. Yardım ağları, muhtemel yardımcı robotları örtülü olarak belirler. Ağ yapısı, merkezi ağ topolojilerine benzer şekilde, bir koordinatör tarafından belirlenir. Son olarak, daha genel bir problem olan, ancak farklı tip ve miktarlarda kaynaklara sahip robotların işbirliği ile gerçekleştirilebilen görevler ele alınmıştır. Bir görev koordinatörü, görevleri düşük maliyet ile yapacak ve yeterli kaynaklara sahip robot koalisyonlarını bulmaktan sorumlu tutulmaktadır. Bu süreç, koalisyon düzenleme oyunları ile modellenmiştir. Kapsamlı benzetim ve robotlar ile yapılan deney sonuçları, önerilen yaklaşımların gerçek zamanlı çoklu robot sistemlerinde pratik olarak uygulanabilirliğini göstermiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	Collective link state
a_i	Link state vector for robot i
B	Channel bandwidth
b	Collective state of all the robots
b_i	Configuration state of robot i
c	Cost parameter
D	Average network density
$E()$	Expected value
E_l	Approximate lower bound
E_u	Approximate upper bound
$\mathcal{F}$	Free configuration space
g	Communication network
$\mathcal{G}$	Set of all possible graphs
g^r	Complete graph
h_i	Goal position of robot i
$\mathcal{K}$	Sequence of networks generated by game moves
l	Path loss factor
$\mathcal{N}_i(g)$	Set of 1-hop neighbors for robot i
$\mathcal{N}_i^+(g)$	Connected set for robot i
$\mathcal{N}_i^-(g)$	Nonconnected set for robot i
N_g	Average number of network updates
N_o	Noise power
$\mathcal{O}_i$	Maximal connected neighbor sets for robot i
$p(\|)$	Conditional probability
P_T	Transmitter power
q_0	Initial distribution
$\mathcal{R}$	Set of robots
r	Number of robots

Q	Index set of robot pairs
$s_i(t)$	Selected mate of robot i for a game move at time t
t_f	Time of task completion
T_g	Network update period
T_{η_i}	Transition matrix for robot i
u_i	Control law
V	Average total payoff
v_i	Individual payoff function
v_{ij}	Pairwise payoff function
x_i	Number of links that need to be broken by robot i
W	Total bandwidth
z_i	Number of desirable, but not attained links for robot i
α	Path loss exponent
δ_{ij}	Distance between robot i and robot j
Δt_g	Game interval
Δt_h	Handling duration
Δt_o	Timeout duration
Δt_{RTT}	Average round-trip time
Δt_t	Task interval
$\mathcal{E}(g)$	Set of edges
η_i	Number of neighbors of robot i
κ	Cost term in payoff function
λ	Expected value of a Poisson distribution
$\xi_i(b)$	Cardinality of maximal connected set
π	Stationary distribution
ρ_i	Radius of robot i
ρ_c	Fixed communication range
ρ_{ij}	Sum of radii of robot i and robot j
ρ_0	Radius of the workspace
ς_{ij}	Communication cost parameter

$\tau_{x_i,n}$	First time to hit state 0 from state n for x_i process
$\chi_i(g)$	Information set

LIST OF ACRONYMS/ABBREVIATIONS

AMCL	Adaptive Monte Carlo Localization
CFG	Coalition Formation Game
CPG	Centralized Pairwise Game
DCOP	Distributed Constraint Optimization Problem
EA	Extended Assignment
IA	Instantaneous Assignment
LBM	Link-Based Mutation
MR	Multi-Robot
MRTA	Multi-Robot Task Allocation
MT	Multiple Tasks
PG	Pairwise Game
PS	Pairwise Stable
RM	Robot mutation
ROS	Robot Operating System
SR	Single Robot
ST	Single Task
TA	Time-extended Assigment

1. INTRODUCTION

This thesis is concerned with multi-robot systems. A multi-robot system is a group of robots that coexist in an environment. Multiple robots are known to have some advantages over single robot systems [1]. First, some tasks may be inherently too complex for a single robot to accomplish, or performance benefits can be gained from using multiple robots. Secondly, building and using several simple robots can be easier, cheaper, more flexible and more fault tolerant than having a single powerful robot. However, multi-robot systems also come with some challenges - some of which are addressed in this thesis.

1.1. Problem Statement

Consider a multi-robot system as seen in Figure 1.1. Suppose the robots are engaged in a task such as patrolling or exploration. Also suppose they encounter dynamic tasks along the way that require cooperative actions. The required cooperation may vary from sole communication to collaborative behavior. Hence, being able to cooperate is a fundamental question. In this thesis, the cooperation problem is examined from three critical and related aspects.

(i) First, the problem of attaining effective communication network topologies is considered. The robots need to augment their individual sensing with the information acquired via communicating with other robots over a network topology. While all-to-all network, as is assumed in most works, is the simplest approach, unfortunately, it faces serious scalability challenges in reliability, interference and security with the growth of robot team size [2]. Furthermore, limited use of communication may be crucial to performance as communication has its own costs in regards to time, power and decision-making. The robot needs to switch its focus from its task to communication and processing of the information received and back again. This problem can be attenuated if each robot is selective in the robots it exchanges information with [3]. Thus, the robots need to have a scheme

Figure 1.1. A multi-robot system of 5 robots in an environment with a variety of obstacles.

that will determine the communication network topology as to ensure the required communication without being fully connected. Furthermore, the network topology needs to be changing as the robots are moving around - in contrast to most work that assumes a fixed topology. The network formation defines what the topology is and how it evolves over time.

(ii) Secondly, the problem of finding an assisting robot is considered. While tasks and their particular requirements will vary depending on the application, in all, the question of which robot should assist which other robot needs to be addressed effectively in order for the tasks to be successfully completed. This is a challenging task. First, as the tasks will occur at unpredictable places or times, the robots cannot be assigned a priori. Secondly, the problem is exacerbated by the fact that some robots will be busy when others are seeking assistance. Finally, there will be task related constraints that need to taken into account.

(iii) Finally, a more general version of this problem is considered. Tasks may require a variety of different resources. This necessitates a group of robots (coalitions) coming together for the task. Again, this is a challenging task. First, the coalitions that are formed should have all the required resources. Secondly, these coalitions need to change as new tasks in different places with different requirements are encountered.

1.2. General Approach

Consider a multi-robot system. The system can be mathematically described by a set of robots $\mathcal{R} = \{1, \ldots, r\}$. It is assumed that each robot $i \in \mathcal{R}$ has radius $\rho_i \in \mathbb{R}$ and is uniquely identifiable. Each robot $i \in \mathcal{R}$ has a time-varying configuration state $b_i(t) \in \mathbb{R}^2$ - assuming a two-dimensional workspace without loss of generality. The time argument is omitted whenever time dependency is clear from the context.

The collective state of all the robots $b \in \mathbb{R}^{2r}$ is defined as $b = \sum_{i \in \mathcal{R}} b_i \otimes e_i$ where e_i are the unit vectors in $\mathbb{R}^r$ and $\otimes$ is the Kronecker product [4]. The index set of robot pairs is defined as $Q = \{ij \,|i, j \in \mathcal{R}, i < j\}$. The cardinality of Q is $|Q| = \binom{r}{2}$. For each pair $ij \in Q$ of robots, their pairwise distance is denoted by $\delta_{ij} = \| b_i - b_j \|$. To be physically feasible, δ_{ij} will satisfy:

$$\forall i, j \in \mathcal{R} \qquad \delta_{ij} \geq \rho_{ij} \tag{1.1}$$

where $\rho_{ij} = \rho_i + \rho_j$ is the sum of their radii. Otherwise, this would imply that the robots are overlapping. Furthermore, if the workspace is bounded by radius ρ_0, then each robot should stay in workspace which implies that:

$$\forall i \in \mathcal{R} \qquad \| b_i \| \leq \rho_{0i} \tag{1.2}$$

where $\rho_{0i} = \rho_0 - \rho_i$. Hence, the free configuration space $\mathcal{F} \subset \mathbb{R}^{2r}$ is defined by Equation 1.1 and Equation 1.2. Elements of feasible configuration space $\mathcal{F}$ constitute the permissible configuration states.

In this setting, multi-robot systems are studied from three different aspects as explained. First, it is argued that effective network topologies can be attained if communication and/or task related objectives are taken into consideration. Hence, the problem is posed as a multi-objective optimization problem and two approaches that enable practical application are proposed. In the first approach, robots decide on their local part of the network topology in a decentralized manner based on pairwise games.

The second is a centralized approach where this job is given to a network coordinator. In this case, the network coordinator's decision making is modeled as a centralized pairwise game. Next, the problem of determining assisting robots in dynamic and cooperative tasks is considered. For this, the concept of assistance networks is developed. Interestingly, the determination of assistance network topology follows the methodology developed for the centralized network formation. In particular, a network coordinator decides on the assistance network topology. Finally, the case of more complex tasks with resource constraints is addressed. Here, again a task coordinator is responsible for finding the group of robots that will be working together to complete a task. The search process is modeled as a coalition game formation where all the tasks and robots' resources are considered.

1.2.1. Communication Network Topologies

The communication network topology is defined by $g(t) \in \mathcal{G}$ where $\mathcal{G} = \{g' | g' \subseteq g^r\}$ is the set of all possible graphs on $\mathcal{R}$ and g^r is the complete graph. The set of edges $\mathcal{E}(g) \subseteq Q$ represents the robot pairs i and j between which a direct edge is established. The immediate neighborhoods $\mathcal{N}_i(g) \stackrel{\triangle}{=} \{j \in \mathcal{R} \,|ij \in \mathcal{E}\}$ correspond to the edges of graph g. It is assumed that the robots can communicate directly with immediate neighbors. Note that the case when $g(t) = g^r$ corresponds to complete network in which all the robots can communicate with each other.

As explained earlier, robots need to be selective as to whom they communicate with. In this perspective, the most common approach is to communicate only with robots within a certain physically range [5–7]. However, distance is not necessarily the optimal selection criteria. The objectives may encode other communication or task-related criteria. In this work, a general setting that enables this is proposed with two alternatives: decentralized and centralized network topologies.

(i) Decentralized Network Topology: In this approach, all the robots individually consider network related objectives and adjust their local network topology intermittently in a decentralized manner. Pairwise games provide a practical and

general scheme for contacting other robots and revising the network topology. A network is deemed acceptable by all the robots using pairwise stability and pairwise Nash equilibrium. The network is updated via pairwise games played among the robots. This is repeated as long as the robots are engaged in their tasks. Pairwise stability is ensured for additive objective functions with symmetric pairwise payoffs.

(ii) Centralized Network Topology: In many applications, the robots' communication related objectives encode neighboring robots related considerations. If decision-making becomes centralized, pairwise stable networks can be attained. A network coordinator becomes responsible for determining the network topology. The robots periodically send their state information to this coordinator. In turn, the coordinator considers the individual payoff functions of all the robots, their current states and the current network simultaneously and finds a network topology acceptable to all the robots. Modeling the network topology formation as a centralized pairwise game, it forms or severs pairwise links based on the improvement the resulting network offers the robot pairs relative to the current network. It is shown that with a particular class of payoff functions, the pairwise game is ensured of convergence to a pairwise stable network.

1.2.2. Assistance Networks

With tasks requiring robot pair cooperative actions, the effective allocation of assisting robots is essential to their successful completion. Most previous works seek to find an explicit assignment of assisting robots - which has proven to be a computationally challenging problem. This problem is addressed by introducing the concept of assistant networks. Assistant networks designate potential helpmates in an implicit manner. Each robot - when faced with a task - attempts to find an assisting robot among its immediate neighbors in the assistance network in a decentralized manner. The assistance network topology is determined by a coordinator with a process model similar to that developed for centralized network topologies using centralized pairwise games.

1.2.3. Coalition Formation Games For Dynamic Multirobot Tasks

Finally, cooperative tasks are generalized by considering tasks that require a set of resources. As the effective formation of robot teams endowed with these resources is crucial, the focus is on effective coalition formation. In this case, a task coordinator - similar to network coordinator – assumes the role of determining coalitions and assigning tasks. Its process is modeled as a coalition formation game where groups of robots are evaluated together in regards to each task's required resources and cost of forming a coalition. As new tasks are encountered, coalitions merge and split - resulting in new coalitions that are capable of doing these tasks.

1.3. Contributions

The major contributions of this thesis can be summarized as follows:

(i) Network Topologies: Two novel approaches to determining network topologies are introduced. These vary in the nature of decision-making - namely decentralized and centralized network topologies.

- Decentralized Network Topology: The network topology evolves in a decentralized manner based on network related payoff functions. In this model, pairwise games provide a practical and general scheme for contacting other robots and revising the network topology. In case of symmetric pairwise payoffs, the network is ensured of being optimal by being pairwise stable.
- Centralized Network Topology: The network topology is determined in a centralized manner via a network coordinator. The process is modeled based on centralized pairwise games. In this case, pairwise payoffs need no longer have to be symmetric for pairwise stability. In particular, with certain pairwise payoffs that encode neighboring robots' related considerations, the network topologies can be shown to be pairwise stable.

(ii) Assistance Networks: A novel concept - referred to as 'assistance networks' - is developed for the problem of tasks requiring the cooperation of robot pairs.

The assistance network designates potential helpmates in an implicit manner. Its topology evolves following a process similar to that developed for the centralized network topology model and is thus ensured of being pairwise stable. As such, as assistance related objectives are optimally satisfied, robots can get assistance or give assistance flexibly.

(iii) Coalition Formation: Coalition formation games are used as in forming robot teams endowed with sufficient resources.

1.4. Organization of the Thesis

The rest of this thesis is organized as follows.

Chapter 2 presents decentralized network topologies. The formation and evolution of the network topology is based on payoff functions and pairwise games where the robots form or sever pairwise links based on the improvement the resulting network offers the robot pairs relative to the current network. In this framework, the equilibrium network is defined based on pairwise stability and pairwise Nash equilibrium. It is shown that in case of mutual link-based network payoff functions, these networks are ensured of being pairwise stable networks that are also pairwise Nash. Extensive simulation results provide insight on performance with respect to a variety of metrics including a comparative study with all-to-all communication.

Next, the case of centralized network topologies is considered in Chapter 3. In this approach, a network coordinator is responsible for determining the network topology. The robots periodically send their state information to the network coordinator. In turn, the coordinator considers the individual payoff functions of all the robots, their current states and the current network simultaneously and finds a network topology acceptable to all the robots. The network topology formation is modeled as a pairwise game. It is shown that with objective functions that encode neighbors' related considerations, each centralized pairwise game converges to a pairwise stable network. Extensive simulations provide statistical results on the resulting network topology and decision-making times including a comparative study with proximity based communi-

cation.

Following, the problem of determining assisting robots in dynamically encountered tasks requiring the cooperation of robot pairs is considered in Chapter 4. Here, the concept of assistance networks is introduced. As explained previously, the assistance network designates potential helpmates in an implicit manner. Its topology is determined by the assistance network coordinator - whose job is to find a network topology acceptable to all the robots - depending on their respective payoff functions. Acceptability is based on pairwise stability and pairwise Nash stability. It is shown that the coordinator can find topologies acceptable to all the robots via playing centralized pairwise games. Extensive simulation results demonstrate that the robots are capable of completing tasks effectively in a variety of different scenarios - including a comparative study with proximity-based assistance. Experimental results with a team of five mobile robots verify the practical applicability of the proposed approach. This has required the design and development of a novel control architecture as explained in detail in Appendix E.

Finally, the problem of forming robot coalitions endowed with tasks' resource requirements is addressed in Chapter 5. In parallel, the definition of tasks is extended to incorporate resource requirements. In this framework, the task coordinator is associated with a process based on coalition formation games. Extensive simulation results demonstrate that the task coordinator is able to find a suitable coalition formation considerably fast.

The thesis concludes with a brief summary and a discussion regarding future work as presented in Chapter 6.

In this thesis, considerable time has been spent on developing the robotic platforms for real-time experiments. In fact, four generations of robots were developed. This endeavor is summarized in Appendix C. Furthermore, the operation of these robots has necessitated the design and development of a novel control architecture. In a collaborative effort, a novel sense-communicate-act architecture has been devel-

oped as presented in Appendix E. Finally, the problem of getting a set of robots in a loosely specified formation - referred to as realization problem - has been studied. In this study, initially an approach based on genetic algorithms has been developed as presented in Appendix F. Due to computational complexity that hinders real-time applications, an alternative approach based on an artificial potential functions has also proposed as presented in Appendix G. Finally, the list of publications is presented in Appendix H.

2. DECENTRALIZED NETWORK TOPOLOGY

In this chapter, the problem of forming and updating the network topology in a multirobot system is studied. Advances in communication networks have enabled coordinated dynamic tasks amongst a group of robots endowed with communication capabilities [8–11]. The robots, while engaged in a particular task, can cooperate via communicating and exchanging their information. However, as the number of communicating robots increases, the reliability of communication is affected by quality fluctuations [2,12–14]. Furthermore, as each robot has to allocate more of its resources away from its task for communication, its task execution will possibly be subject to performance degradation [9]. This can be alleviated by having the robots be selective in the robots they communicate with and thus requiring a lower density communication network [15]. From this perspective, the fundamental question is with which other robots to establish direct links and how to evolve the network topology as robots move around.

To this end, a decentralized model based on network related payoff functions and pairwise games is proposed. A network is deemed acceptable by all the robots using pairwise stability and pairwise Nash equilibrium. As an application, networks that are generated with mutual link-based payoff functions are considered. It is shown that - under some assumptions regarding changes in the configuration states - each game is ensured of converging to a pairwise stable network. This approach is then integrated with a common robotic task where the network is critical to successful task completion. The resulting performance is evaluated with respect to a variety of measures including task completion, network density and the average payoff along with comparative results with all-to-all communication.

The outline of the Chapter is as follows: First, related literature is reviewed in Section 2.1. Next, decentralized network topology is formulated in Section 2.2. The update of network topology based on payoff functions and pairwise games is accomplished as explained in Section 2.3. In this framework, the equilibrium (optimal) network is

defined based on pairwise stability and pairwise Nash equilibrium as explained in Section 2.4. With mutual link-based network payoff functions, the existence of pairwise stable networks is ensured as explained in Section 2.5. Furthermore, these networks are also pairwise Nash stable. Furthermore, each pairwise game converges to a pairwise stable network. The coupling of this approach with a common robotic task (navigation) is presented in Section 2.5.2 where extensive simulation results provide insight on performance with respect to a variety of metrics such as task completion, network density, network change and the average network payoff. The resulting performance is compared with that that of all-to-all communication. The Chapter concludes with a brief summary.

2.1. Related Literature

This problem is related with two areas in the literature: multirobot communication and game theory.

Most multirobot applications assume all-to-all communication [9, 10]. From mobile communication perspective, this can be done either in a broadcast or non-broadcast manner [16]. Broadcast multipoint communication requires a physical medium that connects to all robots where all communication is heard by all [17]. As such, the robots may miss the broadcast information or if multiple robots try to broadcast all at the same time, the resulting network saturation may lead to degradation of quality of the received information [13, 18, 19]. These problems are alleviated in non-broadcast point-to-multipoint or point-to-point communication [20]. However, distributed broadcast scheduling is problematic since it has NP complexity [21]. Thus, with the growth of robot team size, all-to-all communication faces serious scalability challenges [9, 10].

Even if all-to-all communication is achieved, selective communication may be advantageous for performance [15]. This is because communicating robots have an underlying hybrid nature: a continuous aspect pertaining to the robots' physical states and a discrete nature related to communication states [22–27]. Each robot needs to switch its focus between its task and communication intermittently. As the number of

connections increases, it has to devote more of common resources to communication due to increased computation and power requirements [9]. The most common approach to alleviate this problem is to communicate only with robots within a given proximity determined either via on-board sensing or periodic all-to-all communication [5–7]. Another approach is to maintain a sparsely connected communication graph and have virtual all-to-all communication via a routing protocol [28]. Unfortunately, the reliability of the information will be affected by the routing delays. Interestingly, none of the approaches consider communication related objectives [9, 29] and how the network can be formed based on them. For example, resource constrained robots can have other considerations or objectives in regards to what constitutes an acceptable network topology. This issue has been addressed within communication networks assuming stationary agents where the conflict between gain and cost of each connection naturally leads to a noncooperative game theoretic formulation of the problem [12, 30–32]. However, in multirobot settings, the stationarity of the players is no longer valid and as the robots are moving around, the topology of the communication network has to be updated accordingly [33, 34]. Furthermore, it is preferable to have the network evolve in a decentralized manner so that each robot is in complete control of its local part of the network.

2.2. Decentralized Network Topology

The network topology is defined based on the collective link state of the robots $a \in \mathcal{A}$ is defined as $a = \sum_{i \in \mathcal{R}} a_i \otimes e_i$. Here, each $a_{ij}(t) \in \mathcal{B}$ with $\mathcal{B} = \{0, 1\}$ is defined as:

$$a_i(t) = \left[a_{i1}(t) \dots a_{i(i-1)}(t)\, a_{i(i+1)}(t) \dots a_{ir}(t)\right]^T$$

It refers to the link state associated with robot $i \in \mathcal{R}$. Then $a_{ij}(t) = 1$ if and only if robot i chooses a direct link with robot $j \neq i$. Otherwise $a_{ij}(t) = 0$. Note that the states a_{ij} are not necessarily symmetric. Pairwise links are established only with mutual consent – that is a link ij is created if and only if $a_{ij} = a_{ji} = 1$. In the sequel,

the time argument is omitted and a_i is used whenever time dependency is clear from the context.

Each link state $a \in \mathcal{A}$ induces an undirected network $g(a) \in \mathcal{G}$. Let $\mathcal{A} = \underbrace{\mathcal{B}^{r-1} \times \ldots \times \mathcal{B}^{r-1}}_{r \text{ times}}$ denote set of all possible network topologies. The set $\mathcal{G} = \{g' | g' \subseteq g^r\}$ is the set of all possible graphs on $\mathcal{R}$ and g^r is the complete graph. Each graph $g = (\mathcal{R}, \mathcal{E})$ is such that the set of edges $\mathcal{E}(g) \subseteq Q$ represents the robot pairs ij between which there is a direct link. From each robot i's perspective, the corresponding set of links is defined as $\mathcal{E}_i(g) = \{ij \mid j \in \mathcal{N}_i(g)\}$. The case when $g = g^r$ corresponds to all-to-all network - namely every robot has a direct link with every other robot. For each robot i, its neighboring robots $\mathcal{N}_i(g)$ are robots with which direct links exist – namely $\mathcal{N}_i(g) \stackrel{\triangle}{=} \{j \in \mathcal{R} \,|ij \in \mathcal{E}_i(g)\}$.

2.3. Topology Update

The communication network is viewed as composed of agents who are trying to maximize their own benefit from participation in the network. Thus, the updating of the network topology is based on two entities: (i) A set of payoff functions that encode respective benefits quantitatively; (ii) Pairwise games where decisions regarding the network topology are made by utilizing these payoff functions.

2.3.1. Payoff Functions

For each robot $i \in \mathcal{R}$, the payoff is encoded quantitatively by a function $v_i : \mathcal{G} \times \mathcal{F} \to \mathbb{R}$. The payoff function depends on both the link state (and hence the network g) and the configuration state b. This dependency can vary from complete dependency on the network and the state to partial dependency. In our decentralized framework, the dependency is partial, that is, each robot i's payoff function depends on its immediate neighbors $\mathcal{N}_i$ defined by the network g. The actual form of this function can be based on either purely communication related considerations [30, 35] or a combination of communication and task related objectives.

2.3.2. Pairwise Games

Once the robots are all associated with their respective payoff functions, the next issue pertains to the network update based on payoff functions. The update process is modelled as pairwise games. A pairwise game among r robots is an iterated process consisting of a sequence of possibly simultaneous game moves. Its definition is similar to the dynamic process of [36], but differs in that multiple robots are able to remove existing links or add new links concurrently with the restriction that each robot is allowed to participate in only one operation.

Assume that the pairwise game starts with robot configuration state b and the initial topology g. In each game move, each robot i can play at most with one other robot or choose not to play at all. Each robot selects its playing mate individually. The simplest selection scheme is to use a probabilistic scheme with uniform probability. If $s_i(t) \in \mathcal{R}$ denotes robot i's selected partner for a game move at time t, then the probability of robot i selecting robot j:

$$p(s_i(t) = j) = \frac{1}{r} \tag{2.1}$$

In case a robot selects itself ($s_i(t) = i$), this indicates that it is in a "don't care" state. Hence, no synchronization is required for selecting a candidate playing mate.

After a selection is made, the next step is to check whether a game move can be started or not. If there is already a link with the selected robot, this is very simple to do. Otherwise, the robot will attempt to connect temporarily to the selected robot. In either case, the other robot either responds back positively or there is no response. The other robot will respond positively if the selection is mutual or it is in a "don't care" state with only one request for a game move. The no-response case may be attributed to three different reasons. First, selection is not mutual. Secondly, in case a robot gets selected by multiple robots, it will not participate in a game move and thus will not respond. Thirdly it could not connect to the selected robot. A game move starts only if there is a positive response. Hence, each robot i will be engaged in a game move

with another robot with probability:

$$p' = \frac{3}{r^2}(1 - \frac{2}{r})^{r-2} = \frac{3(r-2)^{r-2}}{r^r} \tag{2.2}$$

Thus, at each game move, from the perspective of each robot, the status of only one link can be changed.

Once the robot pairs ij mutually agree to play a game move, the game move proceeds in a manner that is dependent on whether there is an existing link between them, namely whether $ij \in \mathcal{E}(g)$ or not. If the link ij is already in the network g, the decision is to break it up or not. Let $g - ij$ denote the graph obtained by breaking up the link ij from g. Otherwise, there is no connection and the decision is whether to add it or not. Let $g + ij$ denote the graph obtained by adding the link ij to the existing network g. These decisions are based on the payoff functions. Thus, for each pair ij engaged in a move, the link states a_{ij} and a_{ji} are updated as follows:

$$a_{ij} = \begin{cases} 0 & \text{if } v_i(g - ij, b) > v_i(g, b) \\ 1 & \text{if } v_i(g + ij, b) > v_i(g, b) \end{cases} \tag{2.3}$$

$$a_{ji} = \begin{cases} 0 & \text{if } v_j(g - ij, b) > v_j(g, b) \\ 1 & \text{if } v_j(g + ij, b) > v_j(g, b) \end{cases} \tag{2.4}$$

All other entries of the link states a_i and a_j remain unchanged. These actions may leave the link as it is (which may be either no link or a direct link), remove an existing link or setup a new link. A direct link is established iff both $a_{ij} = 1$ and $a_{ji} = 1$. The components of each link state a_i transform throughout the sequence of game moves by all the robots.

The algorithm in Figure 2.1 presents the pairwise game for each robot. All robots apply this algorithm concurrently. The game moves will proceed in an unsynchronized manner. Each pairwise game continues for an interval of Δt_g. During the progress of each game, the robot pairs that are engaged in game moves, but are not currently connected will need to establish temporary pairwise connections in order to exchange

information. However, since each robot can engage at most in one move and some links are already established, there will be only few additional links. However, among these, all those that are not beneficial to both of the robots will be immediately cut-off. Thus, robots do not need to maintain a fully connected network unless it is not absolutely beneficial to all the robots simultaneously. Since all the robots act individually, each pairwise game is decentralized.

```
1: t_s ← t                                          ▷ Note the game start time
2: while t < t_s + Δt_g do
3:     j = s_i(t)                                    ▷ Select playing mate j
4:     if game move btw ij started then
5:         if ij ∈ E_i(g) then                       ▷ Does link ij exist ?
6:             if v_i(g − ij, b) > v_i(g, b) then
7:                 a_ij = 0
8:                 g ← g − ij                        ▷ Break up the link ij
9:                 Send a_ij to robot j
10:            end if
11:        else
12:            if v_i(g + ij, b) > v_i(g, b) then
13:                a_ij = 1
14:                Get a_ji from robot j
15:                if a_ji = 1 then
16:                    g ← g + ij                    ▷ Establish the link ij
17:                end if
18:            end if
19:        end if
20:    end if
21: end while
```

Figure 2.1. Pairwise game for each robot i.

2.3.3. Task & Pairwise Games

The robot team starts with an initial configuration and network topology. The robots are assumed to have their clocks synchronized at the beginning of the task to $t = 0$. Each robot proceeds with its task using information available to it. This will be comprised of its configuration state as well as those of its immediate neighbors as defined by the current network topology. Periodically (with period Δt_t), the robots update the network and thus possibly change network topology. This is achieved via all of them participating in a pairwise game for an interval of Δt_g as specified by the algorithm as shown in Figure 2.1. The pairwise game also provides a general scheme for the robots to contact other robots one by one and decide on the status of respective links. Of course, game period should be considerably smaller than the update period - namely $\Delta t_g < \Delta t_t$. Once the game ends, the network topology remains the same until the next update period. This cycle is repeated as long as the robots are operating.

2.4. Equilibrium Networks

The payoff functions are not sufficient to determine what is an acceptable network for each robot. This is because while each robot has its own individual objective, individual payoffs are coupled with each other. In general, it will not be possible to find a single solution that simultaneously optimizes each objective. Additional preference information regarding what constitutes an acceptable solution is required.

2.4.1. Pairwise Stability

The simplest definition is pairwise stability [37–39]. It is based on two considerations - namely no individual robot has incentive to break up a link and no pair of robots have an incentive to establish a new link. Formally,

Definition 2.1. *(Pairwise stability) : A network g is pairwise stable with respect to v_i, $i \in \mathcal{R}$, if and only if,*

(i) $\forall i \in \mathcal{R}$ *and* $ij \in \mathcal{E}(g)$, $v_i(g,b) \geq v_i(g-ij,b)$ *and* $v_j(g,b) \geq v_j(g-ij,b)$

(ii) $\forall ij \notin \mathcal{E}(g)$, *if* $v_i(g+ij,b) > v_i(g,b)$, *then* $v_j(g+ij,b) < v_j(g,b)$

Pairwise stable networks are robust to one-link deviations by definition. Link creations are promoted by the coordinated initiative of pairs of robots while single robots in isolation decide to remove links.

2.4.2. Pairwise Stable Networks

The existence of pairwise stable networks is not ensured as there may be a cycle of networks [38, 40]. The definition of cycle of networks is based on the concept of improving path. A path is a sequence of networks from one network to another and it should not be confused with a path along links within a given network. In particular, paths where each change offers an improvement to the associated robots are of interest. An improving path is a sequence of networks where the removal or the addition of the link benefits the robot(s) whose consent is necessary for the change [38].

Definition 2.2. *(Improving Path) : An improving path from* g *to* g' *is a sequence of networks* $\{g_1, \ldots, g_N\}$ *with* $g_1 = g$ *and* $g_N = g'$ *such that for any* $k \in \{1, \ldots, N-1\}$ *such that either:*

(i) *If* $g_{k+1} = g - ij$ *for some* ij, *then* $v_i(g_k - ij, b) \geq v_i(g_k, b)$ *or* $v_j(g_k - ij, b) \geq v_j(g_k, b)$

(ii) *If* $g_{k+1} = g + ij$ *for some* ij, *then* $v_i(g_k + ij, b) \geq v_i(g_k, b)$ *and* $v_j(g_k + ij, b) \geq v_j(g_k, b)$

This myopic behaviour can lead to what is known as cycle of networks [38,40]. For example, if two robots add a new link making them better off, this may lead another set of players to delete a link which may leave the first set of players worse off. Of course, if the robots could cooperatively update the network instead of myopic decision making, this may be avoidable. However, in large networks where it is preferable to have each robot be as decentralized as possible, myopic behaviour is natural. The cycle of networks is the counterpart of limit cycles in continuous dynamic systems and

is defined as:

Definition 2.3. *(Cycle of Networks) : A set of networks $\mathcal{C}$ form a cycle if for any $g, g' \in \mathcal{C}$, there exists an improving path from g to g'.*

Hence, in a cycle a number of networks are repeatedly visited. The following lemma states that the improving paths emanating from any starting network must lead to either a pairwise stable network or a cycle.

Lemma 2.4. *[40] For any set of payoff functions v_i, there exist at least one pairwise stable network or closed cycle of networks.*

2.4.3. Pairwise Nash Equilibrium

Another concept is pairwise Nash equilibrium which is refinement of pairwise stability based on Nash equilibrium [41]. A pairwise Nash equilibrium network is immune to the formation of a new link by any two robots and the breaking up any number of links by any individual robot.

Definition 2.5. *(Pairwise Nash Equilibrium) : A network g is pairwise Nash equilibrium with respect to v_i if and only if,*

(i) $\forall i \in \mathcal{R}$ *and* $\forall \ell \subseteq \mathcal{E}_i(g)$, $v_i(g, b) \geq v_i(g - \ell, b)$

(ii) $\forall i, j \in \mathcal{R}$, *if* $ij \notin g$, *then* $v_i(g+ij, b) - v_i(g, b) > 0$ *implies* $v_j(g+ij, b) - v_j(g, b) < 0$.

This is a stronger requirement than the single link robustness check of pairwise stability. While all pairwise Nash equilibrium networks are also pairwise stable, the reverse does not hold in general. A result presented in [42] establishes their equivalence when a simple condition on marginal gains holds.

Definition 2.6. *(Marginal Gain)* $\forall i \in \mathcal{R}$, $\forall g \in \mathcal{G}$, $\forall \ell \subseteq \mathcal{E}_i(g)$, *the marginal gain* $\delta v_i(g, b, \ell)$ *associated with* ℓ *is defined as:*

$$\delta v_i(g, b, \ell) \;=\; v_i(g, b) - v_i(g - \ell, b) \tag{2.5}$$

If the set ℓ consists of only one link - namely $|\ell| = 1$, then it is referred to as link marginal gain. This result is based on $\alpha-$submodularity [42].

Definition 2.7. *($\alpha-$submodularity) : Let $\alpha \geq 0$. The network payoff functions v_i are $\alpha-$submodular in own current links on $\mathcal{G}' \subseteq \mathcal{G}$ iff $\delta v_i(g, b, \ell) \geq \alpha \sum_{ij \in \ell} \delta v_i(g, b, ij)$ for all $g \in \mathcal{G}'$, $i \in \mathcal{R}$ and $\ell \subseteq \mathcal{E}_i(g)$.*

Hence, with $\alpha-$submodularity, marginal gain associated with a group of links already in the network is at least as high as the sum of individual link marginal gains scaled by α. Theorem 2.8 provides the necessary and sufficient conditions on the payoff functions v_i, $i \in \mathcal{R}$, for the set of pairwise stable networks and the set of pairwise-Nash equilibrium networks to coincide.

Theorem 2.8. *[42] The set of pairwise stable and pairwise-Nash equilibrium networks with respect to v coincide if and only if v is α-submodular on the set of pairwise stable networks, for some $\alpha \geq 0$.*

2.4.4. Pairwise Stability, Pairwise Nash & Pareto-Optimal Solutions

Multi-objective optimization is in general different from optimizing with respect to a single global objective function if even the global function is constructed via summing up all the individual objectives. It is known that Nash solutions in noncooperative game settings and Pareto-optimal solutions (namely the solution to a single objective function) are not necessarily the same when the individual objective functions are dependent on other robots' decisions as well [37, 43]. Hence, optimizing with respect to a single objective function - even if done in a decentralized manner via decomposing the global objective function into multiple objectives – will in general have different solutions as compared to multiobjective optimization. Thus, the resulting network may not be efficient since that requires a single objective [44]. Pairwise stable networks are robust to one-link deviations. With Nash equilibrium, from each robot's perspective, it is doing as best it can - given its objective, other robots' configuration states and the current network. As each robot decides for its local part of the network based on

only individual considerations associated with its assigned task [45], a noncooperative game-theoretic approach has lower computational and communication requirements.

2.4.5. Convergence to Pairwise Stable Network

As pairwise games provide a practical scheme for contacting other robots and revising the network topology, it is important to study their convergence properties. Each pairwise game generates a sequence of networks $\mathcal{K}(b) = \{g_k\}_{k=0}^{K}$. The payoff function values will be changing as the robots are moving around. Hence, in general, it is not possible to ensure convergence with changing configuration states. However, if game periods are in general of short duration and if the robots' configuration states are not changing much during the course of the game, the robots can play pairwise games with configuration states fixed to values corresponding to the onset of each game. Even with this assumption, convergence will not be ensured in general. In this section, a set of sufficiency conditions for convergence is presented.

For this, first two discrete-time stochastic processes are defined. Let $\mathcal{O}_i(b)$ denote the maximal connected neighbor sets for robot i at the beginning of a pairwise game. $\mathcal{O}_i(b)$ denotes all the links which pairwise benefit the robot. Namely, there is no other robot $j \in \mathcal{R} \setminus \mathcal{O}_i(b)$ such that $v_i(g_0 + ij, b) > v_i(g_0, b)$ and $v_j(g_0 + ij, b) > v_j(g_0, b)$. Let $\xi_i(b) = |\mathcal{O}_i(b)|$ denote the cardinality of this set. Given a network graph g_k, partition the neighborhood set $\mathcal{N}_i(g_k)$ of each robot i into two subsets as $\mathcal{N}_i(g) = \mathcal{N}_i^+(g) \cup \mathcal{N}_i^-(g_k)$. Neighbors in the connected set $\mathcal{N}_i^+(g_k)$ are robots for which the links will remain connected since $v_i(g_k, b) > v_i(g_k - ij, b)$ and $v_j(g_k, b) > v_j(g_k - ij, b)$. The set of robots with which links need to be established is denoted by $\mathcal{O}_i'(b) = \mathcal{O}_i(b) \setminus \mathcal{N}_i^+(g_k)$. Define $z_i(b, g_k)$

$$z_i(b, g_k) = \xi_i(b) - \left|\mathcal{O}_i(b) \cap \mathcal{N}_i^+(g_k)\right| \tag{2.6}$$

Each $z_i(b, g_k)$ counts the number of desirable, but not currently attained links for robot i under network g_k. Now consider $\{z_i(b, g_k)_{k=1}^{\infty}\}$. It is a discrete-time stochastic process with state space $\{0, \ldots, \xi_i(b)\}$. By definition, at g_k^*, $z_i(b, g^*) = 0$. Neighbors

in the nonconnected set $\mathcal{N}_i^-(g_k)$ are those whose links are better if cut off since either $v_i(g_k, b) < v_i(g_k - ij, b)$ or $v_j(g_k, b) < v_j(g_k - ij, b)$. The set of robots with which links need to be broken is denoted by $\mathcal{O}_i^-(b) = (\mathcal{R} \setminus \mathcal{O}_i(b)) \setminus \mathcal{N}_i^-(g_k)$. Define

$$x_i(b, g_k) = (r - 1 - \xi_i(b)) - \left|(\mathcal{R} \setminus \mathcal{O}_i(b)) \cap \mathcal{N}_i^-(g_k)\right| \tag{2.7}$$

Each $x_i(b, g_k)$ counts the number of links that need to be broken by robot i in association with network g_k. Again, by definition, at g_k^*, $x_i(b, g^*) = 0$. Now, consider $\{x_i(b, g_k)\}_{k=0}^{\infty}$. It is again a discrete-time stochastic process, however now with state space $\{0, \ldots, r - 1 - \xi_i(b)\}$. The following proposition provides sufficiency conditions for convergence.

Proposition 2.9. *If $\forall i \in \mathcal{R}$, $z_i(b, g_k)$ and $x_i(b, g_k)$ are both decreasing functions with respect to k, then $\lim_{k\to\infty} g_k = g^*$ with probability one.*

Proof. As $z_i(b, g_k)$ is a decreasing function, it either decreases by one or stays the same. The value $z_i(b, g_k)$ decreases whenever it starts a game move with a robot that is in the maximal connected set, but with which it does not have direct link. If there are $0 < n \leq \xi_i(b)$ links that still need to be established, then the probability that at least one will be established is expressed as:

$$P(z_i(b, g_{k+1}) = n - 1 \mid z_i(b, g_k) = n) = np'$$

where p' is calculated from Equation 2.2. The probability that the number of to-be-realized links does not change is $1 - np'$:

$$P(z_i(b, g_{k+1}) = n \mid z_i(b, g_k) = n) = 1 - np'$$

Thus, $z_i(b, g_k)$ is a first-order Markov process [46]:

$$\begin{aligned} p(z_i(b, g_{k+1}) = n) &= P(z_i(b, g_{k+1}) = n \mid z_i(b, g_k) = n)p(z_i(b, g_k) = n) \\ &\quad + P(z_i(b, g_{k+1}) = n \mid z_i(b, g_k) = n + 1)p(z_i(b, g_k) = n + 1) \\ &= (1 - np')p(z_i(b, g_k) = n) + ((n + 1)p')\, p(z_i(b, g_k) = n + 1) \end{aligned}$$

If $q_{i,k}(b)$ denotes the distribution vector for robot i with g_k, then $q^T_{i,k+1} = q^T_{i,k} T_{\xi_i}$ where the transition matrix T_{ξ_i} is defined as:

$$T_{\xi_i} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & 0 \\ p' & 1 - p' & \dots & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots & \dots \\ 0 & 0 & \dots & (\xi_i - 1)p' & 1 - (\xi_i - 1)p' & 0 \\ 0 & 0 & \dots & 0 & \xi_i p' & 1 - \xi_i p' \end{bmatrix}$$

with $\xi_i(b)$ written without its argument for simplifying the notation. Since T_{ξ_i} is a lower triangular matrix, the stationary distribution π - namely $\pi^T = \pi^T T_{\xi_i}$ is the unique (up to normalization) left eigenvector associated with eigenvalue 1 of T_{ξ_i}. Starting from any initial distribution $q_{i,0}$, the iteration $q^T_{i,k+1} = q^T_{i,0} T^k_{\xi_i}$ converges to the unique stationary distribution $\pi = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}^T$ with probability one. This in turn implies that $\lim_{k\to\infty} z_i(b, g_k) = 0$ with probability one. A reasoning similar to that applied for $z_i(b, g_k)$ holds for $x_i(b, g_k)$. The process $x_i(b, g_k)$ is a first-order Markov process that converges to the unique stationary distribution with probability one which in turn implies that $\lim_{k\to\infty} x_i(b, g_k) = 0$ with probability one. Hence the result. □

The convergence of the game is achieved when $\forall i \in \mathcal{R}$, $z_i = 0$ and $x_i = 0$. However, as the decision-making is completely decentralized, the robots do not know when it occurs so that they can terminate a pairwise game. Hence, the robots are given *a priori* pairwise game interval Δt_g. The pairwise game interval Δt_g specifies how long each game will be played. It is required that $\Delta t_g < \Delta t_t$. However, if it is too short, then there is a risk that pairwise game terminates before a pairwise stable network is

attained. Hence, its value must be carefully set. A good estimate for Δt_g can be based on the expected number of moves for first hitting state 0 for z_i and x_i. Let $\tau_{z_i,n}$ denote the first time to hit state 0 starting in state n for z_i, namely

$$\tau_{z_i,n} = \min_k z_i(b, g_k) = 0 \text{ where } z_i(b, g_0) = n$$

Let $E(\tau_{z_i,n})$ be its expected value. It is equal to:

$$\begin{aligned} E(\tau_{z_i,n}) &= 1 + \sum_{l=0}^{\xi_i(b)} p(z_i(b, g_1) = l \mid z_i(b, g_0) = n) E(\tau_{z_i,l}) \\ &= 1 + p(z_i(b, g_1) = n \mid z_i(b, g_0) = n) E(\tau_{z_i,n}) \\ &\quad + p(z_i(b, g_1) = n - 1 \mid z_i(b, g_0) = n) E(\tau_{z_i,n-1}) \\ &= 1 + (1 - np') E(\tau_{z_i,n}) + np' E(\tau_{z_i,n-1}) \end{aligned}$$

This leads to the recursive formula:

$$E(\tau_{z_i,n}) = \tfrac{1}{np'} + E(\tau_{z_i,n-1})$$

Noting that $E(\tau_{z_i,0}) = 0$,

$$E(\tau_{z_i,n}) = \frac{1}{p'} \sum_{i=1}^{n} \frac{1}{i} \approx \frac{ln(n) + 0.577}{p'}$$

Now, consider $\tau_{x_i,m}$ denoting the first time to hit state 0 starting in state m for x_i. Using a similar reasoning,

$$E(\tau_{x_i,m}) \approx \frac{ln(m) + 0.577}{p'}$$

An approximate upper bound for E_u for the expected number of moves for convergence can be computed by summing the expected number of moves for first hitting state 0

for z_i and x_i and maximizing:

$$\begin{aligned} E_u &= \arg\max_{n,m} E(\tau_{z_i,n}) + E(\tau_{x_i,m}) \\ &\approx \arg\max_{n,m} \frac{ln(n) + ln(m) + 1.154}{p'} \end{aligned}$$

The maximal value occurs at $m = (r-1)/2$ and $n = (r-1)/2$. This corresponds to the case when during the course of the pairwise game, game moves are such that first all connections that need to be established are actually set up followed by breaking up all connections that need to be broken. In applications, establishing and breaking links will be intertwined. A lower bound for the expected number of game moves E_l can be calculated as:

$$E_l = \frac{1.154}{p'}$$

With an average round-trip time of Δt_{RTT} seconds per communication related message, the expected game interval is between $E_l \Delta t_{RTT}$ and $E_u \Delta t_{RTT}$ seconds. This is the average number of moves. In order to handle outlier cases, the game interval will need to be extended based on E_u.

2.5. Application

This section presents an application of the proposed approach in network formation. As different payoff functions will admit different pairwise stable networks or possible a cycle of networks, the existence of pairwise stable networks can be established depending on the particular form of payoff functions. The form of the payoff is restricted by the decentralized nature of decision-making - as the robots should be capable of making decisions using pairwise contacts. In particular, mutual link-based payoff functions are considered [47]. Each payoff function is defined as the sum of pairwise payoffs with immediate neighbors as: $v_{ij} : \mathcal{G} \times \mathcal{F} \to \mathbb{R}$ with immediate neigh-

bors [37, 48]:

$$v_i(g, b) = \sum_{j \in N_i(g)} v_{ij}(g, b) \tag{2.8}$$

Consequently, the individual payoff functions $v_i(g, b)$ measure the overall satisfaction with the respective $\mathcal{N}_i(g)$. Furthermore, each pairwise payoff function v_{ij} is assumed to have a form as follows:

$$v_{ij}(g, b) = v_{ji}(g, b) = \delta v(ij, b) = \delta v(ji, b) \tag{2.9}$$

Benefits accrue only from direct links and each link yields a fixed benefit or loss. Admittedly, this is a simple payoff and the existence of pairwise stable networks is an immediate consequence of its form. Furthermore, pairwise stable networks are also pairwise Nash equilibrium networks as shown in Proposition 2.10.

Proposition 2.10. *Let v_i, $i \in \mathcal{R}$, be a set of mutual link-based payoff functions. All pairwise stable networks are also pairwise Nash equilibria.*

Proof. Consider the marginal gain $\delta v_i(g, b, \ell)$ associated with ℓ. By construction, it is equal to

$$\delta v_i(g, b, \ell) = \sum_{j \in \mathcal{N}_i(g)} v_{ij}(g, b) - \sum_{j \in \mathcal{N}_i(g-\ell, b)} v_{ij}(g - \ell, b) \tag{2.10}$$

Expand the right hand side of Equation 2.10

$$= \sum_{ij \in \ell} \left(\sum_{k \in \mathcal{N}_i(g)} v_{ik}(g, b) - \sum_{k \in \mathcal{N}_i(g-ij)} v_{ik}(g - ij, b) \right)$$

Using mutual link-based property, this is equally expressed as:

$$= \sum_{ij \in \ell} \left(\sum_{k \in \mathcal{N}_i(g)} v_{ik}(ik, b) - \sum_{k \in \mathcal{N}_i(g-ij)} v_{ik}(ik, b) \right)$$

After some manipulation

$$\delta v_i(g, b, \ell) \;=\; \sum_{ij \in \ell} \delta v_i(g, b, ij)$$

Since this holds $\forall i \in \mathcal{R}$,$\forall g \in \mathcal{G}$, $\forall \ell \subseteq \mathcal{E}_i(g)$, α-submodularity with $\alpha = 1$ is satisfied. By Theorem 2.8 [42], this implies that pairwise stability and pairwise Nash stability coincide, hence the result. □

The network game framework, while relevant, may seem unnecessary in such a simple case. However, in practice this is not the case due to decentralized nature of decision-making. As the robots cannot be in communication all at the same time, they have to follow a certain scheme for contacting the other robots - regardless of the form of payoff functions. The pairwise game setting provides such a practical scheme that can be simultaneously utilized by the robots. The robots - via the pairwise games - contact other robots one by one and determine their local network topology depending on the exchanged information. In this case, each pairwise game is ensured of converging to a pairwise stable network with probability one as shown in Proposition 2.11.

Proposition 2.11. *If $\forall i \in \mathcal{R}$, v_i is mutual link-based payoff function with a fixed value of b and g^* is a pairwise stable network, then $\lim_{k\to\infty} g_k = g^*$ with probability one.*

Proof. If v_i is a mutual link-based payoff function , then $\forall j \in \mathcal{O}_i'(b)$, $\delta v_{ij}(g_k, b) > 0$. Hence, a link –once established - remains so. Thus $z_i(b, g_k)$ is a decreasing function with respect to k. Similarly, $\forall j \in \mathcal{O}_i^-(b)$, $\delta v_{ij}(g_k, b) < 0$. Hence, a link, once removed, remains so. Thus $x_i(b, g_k)$ is a decreasing function with respect to k. As $x_i(b, g_k)$ are both decreasing functions with respect to k, the results follows from Proposition 2.9. □

2.5.1. Pairwise Stable Networks

The topology of pairwise stable network formed is investigated via a series of simulations with 20 robots located in a workspace of about 12.5 square kilometers. We consider robot configurations varying in packedness (low, medium, high) and for various communication costs (low, medium, high) and investigate the resulting pairwise stable networks. Payoff functions are assumed to be mutual link-based payoff functions where each v_{ij} encodes the trade-off that exists among competing factors - namely the degree of satisfaction and the cost incurred for having this link [31, 49] as:

$$v_{ij}(g,b) = B\log_2\left(1+\frac{W}{B}SNR_{ij}\right) - \varsigma_{ij}\frac{P_T}{l(\delta_{ij})} \tag{2.11}$$

As formed links can sustain an information flow that is proportional to link quality [50], it is reasonable to require that pairwise interactions should occur only if the link quality is acceptable. Link quality is commonly assessed by signal to noise ratio or related measures [13, 18, 19, 50, 51]. The first term is the capacity of the channel given by the Shannon's well-known formula based on signal to noise ratio SNR_{ij} where B is the channel bandwidth and W is the total bandwidth [35]. The higher its value is, more reliable is the link. As SNR_{ij} is a decreasing function of pairwise distance – namely $SNR_{ij} \propto \frac{1}{\delta_{ij}}$, the gain of communication decreases with increasing distance δ_{ij}. The second term measures the cost of establishing this link with a fixed amount of transmitter power P_T per channel of communication and $l(\delta_{ij}) = \frac{0.097}{\delta_{ij}^{\alpha}}$ denotes the path loss factor due to diffusion and absorption in the environment. Here, α is the path loss exponent and ς_{ij} is the cost parameter [52]. The lower is its value, better it is for the robots. The cost of communication increases as the distance δ_{ij} increases. Let it be noted a form similar to that of Equation 2.11 is also obtained when the payoffs are based on signal-to-interference-plus-noise ratio (SINR) [30, 53]. As MAC protocols ensure that interference from different nodes is minimized, co-channel interference can be neglected [54]. The communication payoff parameters as given in Table 2.1 are adapted from [52] and correspond to physically realistic settings. Noise power is used in accordance with outdoor environment [55]. Path loss exponent α is also selected for

Table 2.1. Communication payoff parameters.

Parameter	**Value**
Transmitter power P_T	1 W
Total bandwidth W	10^6 Hz
Signal bandwidth B	10^4 Hz
Noise power N_0	5×10^{-15} W
Path loss exponent α	2.5
Cost parameter ς_{ij}	0.1

the outdoor environment.

The resulting networks are as shown in Figure 2.2. It is observed that as packedness increases, so does the connectivity of the resulting network. This implies that the closer the robots are, they will be more likely to be connected and thus be cooperating. Furthermore, as communication cost increases, the resulting network connectivity decreases.

2.5.2. Coupling with Robotic Task

In this section, the proposed approach is coupled with multirobot navigation which is a common robotic task [56–58]. The robots all have cylindrical bodies with radius $\rho_i = 0.25$ meters and are deployed in a confined area of radius $\rho_0 = 2$ km where the task of each robot $i \in \mathcal{R}$ is to navigate simultaneously to its a priori specified goal position $h_i \in \mathbb{R}^2$ without any collisions along the way. The details of robot dynamics are presented in Appendix A. A task is completed successfully iff all the robots reach their target positions in the workspace – namely $\forall i \in \mathcal{R}$, $b_i^* = h_i$. The task terminates and is considered to be unsuccessful in case of any collisions or if the robots stop moving without getting to their goal positions. Hence, the robots need other robots' configuration state information as there is a high risk of collisions or blocking each other. As specified, this problem is not a formation control problem as there are no

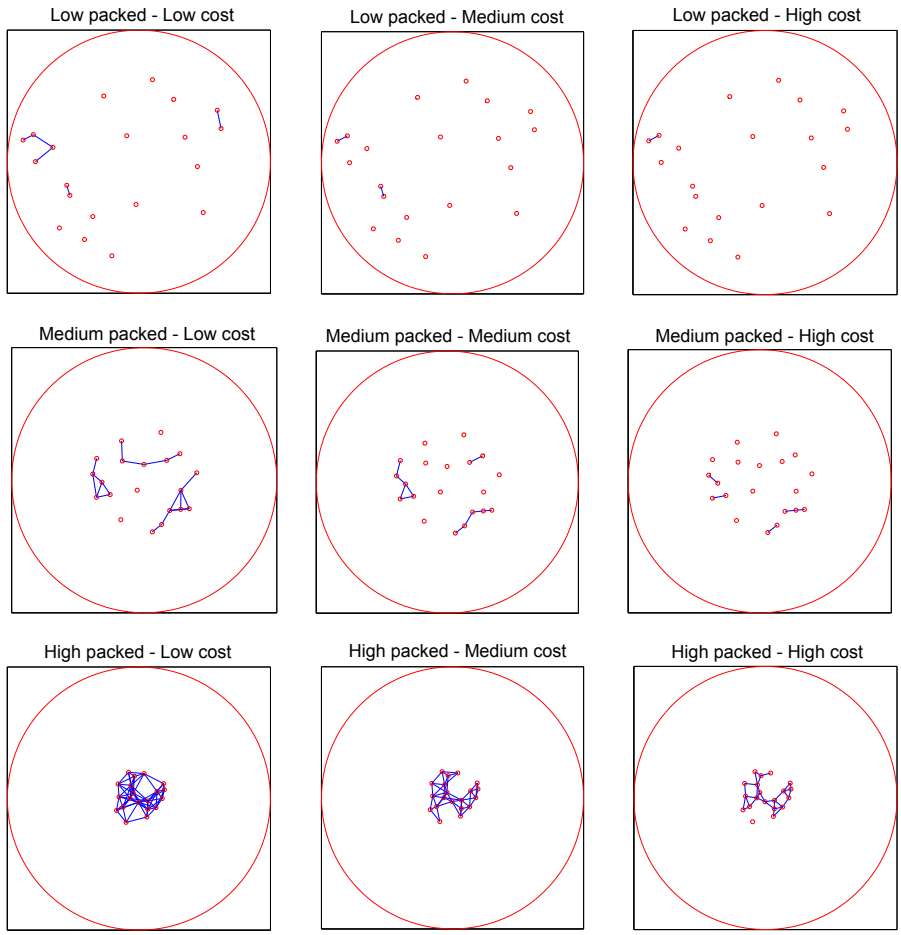


Figure 2.2. Pairwise stable networks for robot configurations varying in packedness (low, medium, high) and communication costs (low, medium, high) in a workspace of 12.5 square kilometers.

restrictions on the relative poses of the robots during the course of their task.

First, the coupling between update periods is studies with Δt_t varying from 2.5 minutes to 10 minutes in order determine a suitable pairwise game interval Δt_g for each task interval respectively. The robots are assumed to move with arbitrary velocities with maximum speed of 0.3 meters/second. In order to determine an appropriate value, 50 simulations are performed for each task interval where all the pairwise games are played until pairwise stable networks are reached. As discussed previously, it is required that $\Delta t_g < \Delta t_t$. In particular six different game intervals Δt_g are considered. The first two are based on the expected number of game moves as computed in Section 2.4.5. For

20 robots, the expected number of game moves has a theoretical expected lower bound $E_l = 1025$ and an expected upper bound $E_u = 5022$ moves that can be computed as explained in Section 2.4.5. With an average round-trip time of $\Delta t_{RTT} = 4.4$ milliseconds per communication related message passing [59], the expected value of Δt_g is between 4.5 and 22 seconds. For each Δt_g , the percentage of pairwise games that have attained pairwise stable networks within this period are determined. The results are presented in Table 2.2. It is observed that with $\Delta t_t = 2.5$ minutes and $\Delta t_g = 22$ seconds, 97% of the resulting networks are pairwise stable. With this Δt_g, as task interval Δt_t is increased, the resulting networks are less likely to be pairwise stable which implies that the communication payoffs are not as optimized as they can be. Thus, the resulting networks are not as satisfactory as they can be. This is expected since with greater task intervals Δt_t, the configuration states will change to a greater extent which will possibly require the network to change more. Hence, higher Δt_t intervals require longer time allocated for network related decision making. The game interval Δt_g is then increased successively in order to have a pairwise stable network percentage of 99.8% for each Δt_t. Finally, with $\Delta t_g = 46$ seconds, all the networks become pairwise stable. This value is about twice the upper bound for the expected value of Δt_g.

Table 2.2. The percentage of pairwise games that reach pairwise stable networks in 20-robot team.

Δt_t (minutes)		2.5	5	10
Δt_g (seconds)	4.5	26.7	12.3	4.5
	22	97.4	91.1	78.6
	33	99.8	99.2	97.3
	37	100	99.8	98.9
	42	100	100	99.8
	46	100	100	100

Next, random goal configurations varying in packedness (low, medium and high) and communication costs ς_{ij} (low=0.025, medium=0.05 and high=0.1) are considered.

Table 2.3. Simulation results for robot configurations varying in packedness and communication costs for pairwise game in 20-robot team.

Packedness	L			M			H		
Comm. Cost ς_{ij}	L	M	H	L	M	H	L	M	H
Task Completion %	100	100	100	100	100	100	100	100	100
Avg. Network Density $\bar{D}$	3.4	1.85	1.2	7.9	4.5	2.5	18	11.8	7.2
Avg. Task Comp. Time t_f (min.)	194	194	193	161	159	158	144	143	142
Avg. Network Change $\bar{C}$	0.6	0.4	0.3	1.2	0.8	0.6	1.9	1.6	1.2
Avg. # of Network Updates N_g	70.4	70.3	70	58.3	57.8	57.3	52.1	51.7	51.4
Avg. Total Payoff $\bar{V}$ ($\times 10^7$)	0.21	0.12	0.07	0.51	0.29	0.15	1.35	0.84	0.49

Table 2.4. Simulation results for robot configurations varying in packedness and communication costs for all-to-all communication in 20-robot team.

Packedness	L	M	H
Task Completion %	100	100	100
Avg. Network Density $\bar{D}$	100	100	100
Avg. Task Comp. Time t_f (min.)	198.4	162.2	149.8
Avg. Network Change $\bar{C}$	0	0	0
Avg. # of Network Updates N_g	-	-	-
Avg. Total Payoff $\bar{V}$ ($\times 10^7$)	-2906	-1782	-1235

For each pair of goal packedness and communication cost levels, the simulations are repeated 50 times with random initial positions. Δt_t and Δt_g are set to 2.5 minutes and 33 seconds, respectively.

The results are presented in Table 2.3 for pairwise game based network and in Table 2.4 for all-to-all network by using various measures. For this purpose, a variety of performance metrics is used in the simulations.

- Task completion percentage: It is the percentage of tasks that have been successfully completed. Task completion is determined via checking $b(t_f) = h$. As explained earlier, any collision among the robots during task execution or their stopping without getting to their goal positions results in the task being unsuccessful. Average task completion time is computed based on tasks that are successfully completed.
- Average network density $\bar{D}$: The network density D is a measure of network connectivity. Let $t'_l = l(\Delta t_t + \Delta t_g)$ denote the time at which l th pairwise game is finished.

$$D = 100\frac{1}{N_g r(r-1)} \sum_{l=1}^{N_g} \sum_{i \in \mathcal{R}} |\mathcal{N}_i(g(a(t'_l)|$$

If $D = 100$, the network has all-to-all connection and the number of links is $\frac{r(r-1)}{2}$. With $D < 100$, the number of links goes down to D percent of $\frac{r(r-1)}{2}$. Hence, comparable task completion rates can be achieved simultaneously with low connectivity. Average network density $\bar{D}$ is computed over all the simulations with successful task termination.

- Average network change $\bar{C}$: The network change percentage C is the time average of the change percentage of the network defined.

$$C = \frac{2}{N_g r(r-1)} \sum_{l=1}^{N_g} \left|\mathcal{E}(g(a(t'_l))) - \mathcal{E}(g(a(t'_{l-1})))\right| + \left|\mathcal{E}(g(a(t'_{l-1})) - \mathcal{E}(g(a(t'_l)))\right|$$

Again, average network change $\bar{C}$ is computed over all the simulations with successful task termination.

- Average total payoff: Average total payoff V for a simulation is defined as the average of sum of individual payoffs over all the networks formed during a task:

$$V = \frac{1}{N_g} \sum_{l=1}^{N_g} \sum_{i \in \mathcal{R}} v_i(b(t'_l), g(a(t'_l)))$$

Average total payoff $\bar{V}$ is computed over all the simulations with successful task

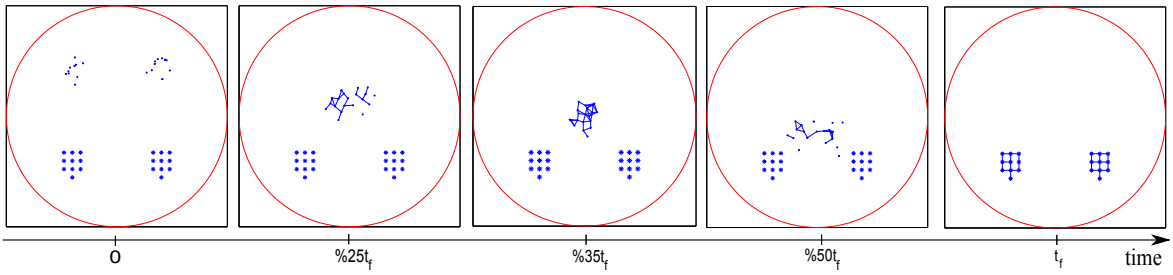


Figure 2.3. A sampled network evolution for $\Delta t_t = 2.5$ minutes in 20-robot team. In each graph, the open circles indicate b_i and star marks indicate the goal positions. The first figure presents a random initial positions while the remaining four show robots' configuration and the communication network at four stages of the task. The task completion time t_f is 243 minutes. The number of pairwise games N_g is 79.

termination.

For all cases, the tasks are completed successfully, but the proposed approach achieves this with the average network density less than 12%, which lessens considerably the communication and computation complexities. Although the proposed approach requires extra computation to determine the pairwise stable network, the task completion time is less than that of all-to-all communication. Increase in the communication cost ς_{ij} leads to the network with less connectivity as expected.

Finally, the robots engage in a task where the goal positions h_i are clustered into two groups of 10 robots in a rectangular grid topology with $\rho_d = 150$ meters between two neighboring positions as shown in Figure 2.3 . The initial configurations are also clustered in two groups of 10 robots where the robots in each group are randomly positioned within an area of 0.3 square kilometers and are located diagonally away from their goal positions as shown in the same figure. The goal positions of robots in the left group are within the right goal area while those of the right group are within the left goal area as shown in the same figure. At the start of a navigation task, the robots are fully disconnected. First, these simulations are repeated 50 times with arbitrary initial configurations without a network among the robots. As expected,

when the robots are not communicating, collisions and blockings occur. Successful task completion percentage is less than 2%. The same task is then considered with the proposed approach. A sample task with initial and goal configurations at $t = 0$ as shown in Figure 2.3 with $\Delta t_t = 2.5$ minutes. The task is successfully completed after $t_f = 243$ minutes with the network updated $N_g = 79$ times. The evolution of the communication network at four sampled instances from a sample task is as shown in remaining part of Figure 2.3. It is observed that throughout the task, while the communication network is relatively sparse, the robots are able to complete their given task. The simulations are then repeated again 50 times with for each task interval Δt_t with game period Δt_g selected as to have 99.8% pairwise stability. For $\Delta t_t = 2.5$ minutes, $\Delta t_g = 33$ seconds while for $\Delta t_t = 10$ minutes, $\Delta t_g = 42$ seconds. Thus, game intervals are considerably smaller than the update intervals as required. The results are presented in Table 2.5. The distance moved between updates increases from an average of 36 meters for 2.5 minutes to about 144 meters for 10 minutes. The robots are assumed to be starting at random initial configurations as explained previously with the initial network fully disconnected.

Finally, the case of fully connected network is connected. It is known that in this case, the task completion is 100%. It is observed that with update period $\Delta t_t = 2.5$ minutes, the robots are able to accomplish their tasks with 100% completion percentage. As update periods increase, task completion percentages are still very high. However, compared to all-to-all communication, it is observed that they are slightly lower as expected. This is because the robots are moving around by greater amounts while communication network is not updated, information sets may become insufficient and the probability of collisions increase. On the other hand, it is observed that pairwise games based network updating also enables major improvements within two aspects of task accomplishment. First, average network density varies around 10% and 12% which indicates that the communication network is relatively sparse regardless of the update period. This is a major improvement in comparison to all-to-all communication with 100% network density as it means nearly 90% less of the resources will be allocated to communication. Secondly, surprisingly, average task completion time is comparatively shorter as compared to all-to-all communication. This can be

attributed to the fact that the robots have to take a much smaller number of robots into account while updating their motion control laws and consequently are wandering around less. As expected, as update period increases, so does the network change between consecutive updates. With an update period $\Delta t_t = 10$ minutes, average network change percentage is greater than those of $\Delta t_t = 2.5$ min and $\Delta t_t = 5$ minutes. The last row of Table 2.5 gives comparative results with respect to average total payoff. It is observed that while pairwise game networks have positive communication payoffs, this is not the case for all-to-all communication. This means that while all the links in the pairwise games based communication network are ensured of good quality, this does not necessarily hold for those in all-to-all communication. Hence, the information from some links will possibly be unreliable. Of course, any usage of unreliable information will adversely affect successful task completion rates. This does not come up in our simulations as it is assumed that all the received information is reliable.

Table 2.5. Simulation results for pairwise game (PG) and all-to-all (A) communication networks in 20-robot team.

Network	PG			A
Δt_t (minutes)	2.5	5	10	-
Δt_g (seconds)	33	37	42	-
Avg. Network Density $\bar{D}$	12.4	11.9	10.8	100
Task Completion %	100	98	88	100
Avg. Task Completion Time t_f (minutes)	215.2	200.1	186.3	265.5
Avg. Network Change $\bar{C}$	2.8	5.1	11.7	0
Avg. Number of Network Updates N_g	70	35.1	16.8	-
Avg. Total Payoff $\bar{V}$ ($\times 10^7$)	0.69	0.6	0.28	-5971

The decentralization of decision-making comes with the issue of convergence. Since the robots are only aware of their immediate neighbors, with the increase in the number robots the decision-making takes longer time to converge. Now, the case of 100 robots is considered. Their initial and goal positions are defined similar to that

of 20 robots. Medium level of communication cost ς_{ij} is selected. Since the number of robots is increased, the environment will change much. Hence, the update period Δt_t is set to 1 minute. The game period Δt_g is set to 33 seconds as in the case of 20 robots, which speeds up the decision-making but may lead to the non-stable networks. At the start of a navigation task, the robots are fully disconnected. The simulations are repeated 50 times with arbitrary initial configurations. The results are presented in Table 2.6. It is observed that with considerably low network density, high percentage of task completion is obtained. However, the resulting networks after the decision-making have some undesired and desired links in order to converge to a pairwise stable network. While average number of links which are desired to be established is about 10 links per game, average number of links which are desired to break up is about 8 link per game. To improve the convergence, the game period can be increased at the first game. Since the changes between the consecutive games will be relatively small, the game period can decreased in the next games.

Table 2.6. Simulation results for pairwise game (PG) communication networks in 100-robot team.

Avg. Network Density D	6.7
Task Completion %	96
Avg. Task Completion Time t_f (minutes)	382
Avg. Network Change C	0.2
Avg. Number of Network Updates N_g	246
Avg. Total Payoff V ($\times 10^7$)	10.9

2.6. Conclusion

This chapter studies the problem of communication network formation in a multirobot system and proposes a decentralized model of how the network may form and evolve over time based on network related payoff functions and pairwise games. The robots update the network topology periodically where the game interval is consider-

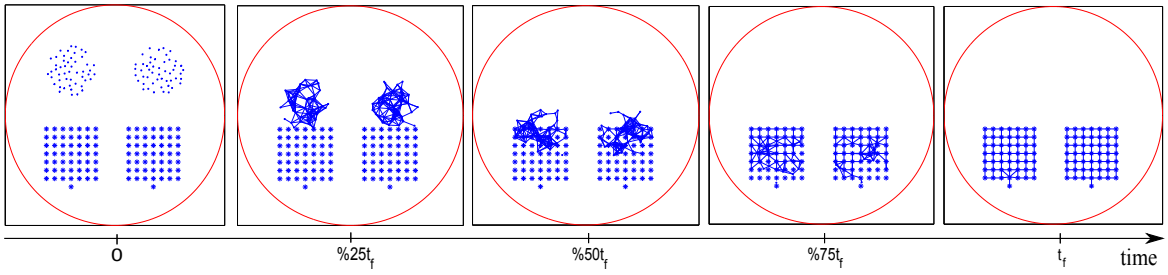


Figure 2.4. A sampled network evolution for $\Delta t_t = 1$ minute in 100-robot team. The task completion time t_f is 355 minutes. The number of pairwise games N_g is 59 for $0.25t_f$, 118 for 0.5 t_f, 177 for 0.75 t_f and 236 for the whole task.

ably smaller than the update period. The equilibrium networks are defined based on pairwise stability and pairwise Nash equilibrium. The pairwise game setting provides a general and practical scheme that can be employed by the robots simultaneously in order to contact other robots and update their local network topology depending on the exchanged information. As an application, the case of mutual link-based payoff functions is considered. It is shown that if each game is played with configuration states fixed to values corresponding to the game onset, the resulting networks are pairwise stable and thus acceptable to all the robots. The proposed approach is coupled with a common robotic task (navigation) where the network is critical to successful task completion. The results indicate that the task can be accomplished with relatively much lower network density in shorter time as compared to all-to-all communication. As such, pairwise games provide a practical and general scheme for contacting other robots and revising the network topology.

3. CENTRALIZED NETWORK TOPOLOGY

This chapter continues to study the problem of forming and updating the network topology. However, differing from the previous chapter, a centralized model is proposed. This model is based on network related payoff functions. However, in contrast to decentralized decision-making, now a network coordinator is held responsible for finding a network topology that is acceptable to all the robots. Its process is now modeled as a centralized pairwise game. While decision-making becomes centralized, the optimality of resulting network topologies can be ensured with less restrictive payoff functions. In particular, with payoff functions that satisfy local spillovers, convexity and strategic substitutes properties, each centralized pairwise game is ensured of convergence to a pairwise stable network.

The outline of the Chapter is as follows[1] : In Section 3.1, the role of the network coordinator with respect to the network is explained. The concept of acceptability is based on payoff functions, pairwise stability and pairwise Nash stability in Section 3.2. It is shown that the coordinator can find topologies acceptable to all the robots via playing centralized pairwise games as explained in Section 3.3. The simulation results in Section 3.4 provide comparative studies and insights on the resulting network topology and the average number of game moves. The Chapter concludes with a brief summary.

3.1. Network Coordinator

A network coordinator is responsible for determining the network topology[2] . It updates the network every T_g seconds. Periodically, the robots send their current configuration states to the network coordinator as seen in Figure 3.1b. If the robots synchronize their clocks at the beginning of the task as is assumed in most multirobot settings [60–62], then these are done concurrently; otherwise they are done asynchronously. Here, the robots are assumed to have their clocks synchronized at the

[1]For related literature, the reader is referred to Section 2.1.

[2]This may be one of the robots with the additional task of being a network coordinator.

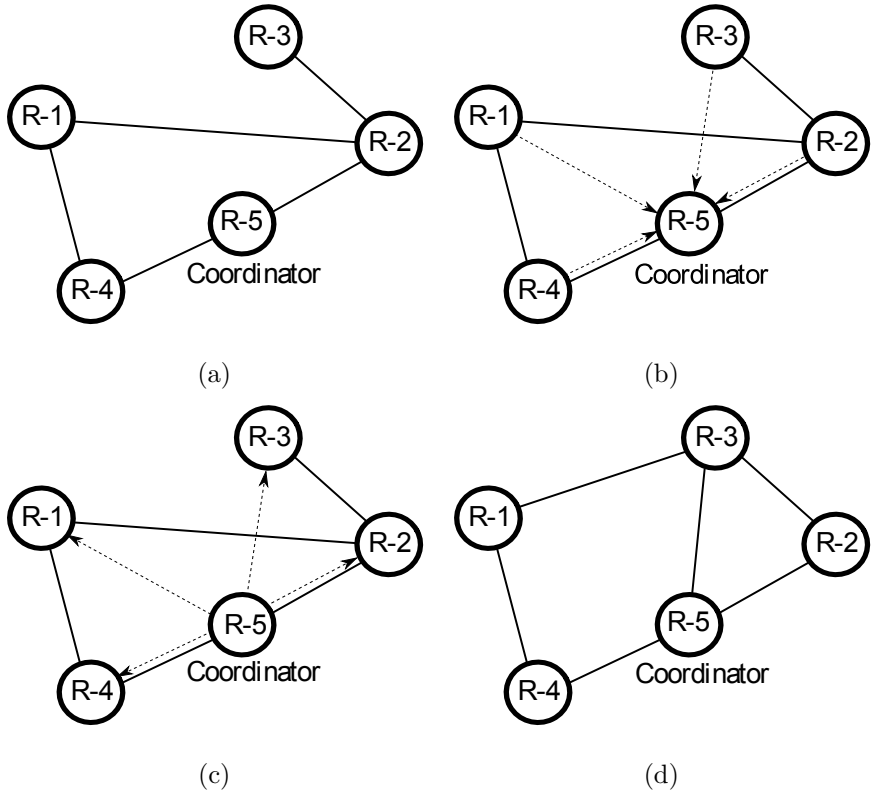


Figure 3.1. Robot-5 is acting as the network coordinator. Solid edges represent the network topology. One-way dashed edges show the temporary communications. (a) Initial network topology; (b) Periodically, all the robots send their information to the coordinator; (c) In turn, the coordinator sends the network information to the robots; (d) Final network topology.

beginning of the task to $t = 0$. In turn, the network coordinator determines a new topology and transmits this to all the robots as also seen in Figure 3.1c. The network $g(t)$ remains the same until it is changed by the network coordinator.

The network coordinator designates the network topology by a time-varying state $a \in \mathcal{A}$ as:

$$a = \left[a_{12}, a_{13} \ldots a_{i(i+1)}\, a_{i(i+2)} \ldots a_{(r-1)r}\right]^T \tag{3.1}$$

where $a_{ij} \in \mathcal{B}$ with $\mathcal{B} = \{0, 1\}$ and $\mathcal{A} = \underbrace{\mathcal{B} \times \ldots \times \mathcal{B}}_{\binom{r}{2} \text{ times}}$. In case $a_{ij} = 1$, robot i and

robot j are directly connected. Otherwise there is no direct edge between robots i and j. The topology of network g is defined by a as:

$$\mathcal{E}(g) \;=\; \{ij \,|\, a_{ij} = 1\} \tag{3.2}$$

The 1-hop neighborhoods $\mathcal{N}_i(g) \stackrel{\triangle}{=} \{j \in \mathcal{R} \,|ij \in \mathcal{E}\}$ correspond to the edges of graph g. The cardinality of $\mathcal{N}_i(g)$ is denoted by $\eta_i(g)$. The topology can vary from being fully connected ($\eta_i(g) = r - 1$, $\forall i \in \mathcal{R}$) to completely disconnected ($\eta_i(g) = 0$).

3.2. Centralized Network Topology

The coordinator needs to find a topology that is acceptable to all the robots. Acceptability is defined based on payoff functions encoding the robots' assistance related preferences individually. A simple criterion is to use proximity so that only robots within a certain range are able to assist each other [5–7]. However, distance may not always be best criterion in determining the admissible robots. They may have other individual preferences regarding what their neighborhood $\mathcal{N}_i(g)$ should be. From this perspective, each robot's preferences are related with the gain and cost associated to having a particular neighborhood $\mathcal{N}_i(g)$. Correspondingly, the payoff functions $v_i : \mathcal{G} \times \mathcal{F} \to \mathbb{R}$ are comprised of two terms encoding gain and cost respectively as:

$$v_i(g, b) = \pi_i(g, b) - \kappa_i(g) \tag{3.3}$$

Each term depends only on itself and its neighbors $\mathcal{N}_i(g)$.

The gain $\pi_i(g, b)$ is related with when the robot expects to do better in regards to the network. Individually, it prefers to communicate with as many robots as possible as this means more information regarding other robots. At the same time, it prefers neighbors that are themselves communicating with any few other robots as possible as response time from each will deteriorate as its connections increase. These

considerations are encoded by a gain function as:

$$\pi_i(g, b) = \Psi_{1,i}(\eta_i(g)) + \sum_{j \in \mathcal{N}_i(g)} \Psi_{2,ij}(\eta_j(g), b) \tag{3.4}$$

Payoffs that depend on the identity of neighbors and distribution of links of all the other robots as given in Equation 3.4 are said to have local spillovers property [63].

The functions $\Psi_{1,i}$ and $\Psi_{2,ij}$ can be arbitrarily chosen as long as $\Psi_{1,i}$ increases in the number of its collaborative links and $\Psi_{2,ij}$ decreases in the number of its neighbors' links. Here, the following particular forms are assumed:

$$\begin{aligned} \Psi_{1,i}(\eta_i(g), b) &= \eta_i(g) \frac{\rho_0}{\sum_{j \in \mathcal{N}_i(g)} \delta_{ij}/\eta_i(g)} \\ \Psi_{2,ij}(\eta_j(g), b) &= \frac{1}{\eta_j(g)} \frac{\rho_0}{\delta_{ij}} \end{aligned}$$

Recall that $\delta_{ij} = \| b_i - b_j \|$.

As such, both $\Psi_{1,i}$ and $\Psi_{2,ij}$ terms decrease in the average distance between its neighbors. Under this kind of payoff structure, a robot will connect with as many robots as possible which are as near to itself as possible and have themselves few links.

This dependence can be quantitatively characterized by convexity and strategic substitutes properties [63, 64]. Convexity means that marginal return of robot i from an edge is increasing in the number of edges it has formed. Formally, $\pi_i(g, b)$ is convex iff $\Delta\Psi_1(x) > 0$ where $\Delta\Psi_1(x) = \Psi_1(x) - \Psi_1(x-1)$ defines the marginal return of robot i from Ψ_1. Strategic substitutes property means that marginal return of a robot i from an edge with a robot j is decreasing in the number of edges formed by robot j. Again, formally $\pi_i(g, b)$ has strategic substitutes property iff $\Delta\Psi_2(x, b) < 0$ where $\Delta\Psi_2(x, b) = \Psi_2(x, b) - \Psi_2(x-1, b)$ defines the marginal returns of robot i from Ψ_2.

Proposition 3.1. *The function $\pi_i(g, b)$ in Equation 3.4 under the assumed forms of $\Psi_{1,i}$ and $\Psi_{2,ij}$ is convex and has strategic substitutes property.*

Proof. Note $\Psi_{1,i}(\eta_i(g), b) = \eta_i(g)\frac{\rho_0}{\sum_{j\in\mathcal{N}_i(g)}\delta_{ij}/\eta_i(g)}$ and $\Psi_{2,ij}(\eta_j(g), b) = \frac{1}{\eta_j(g)}\frac{\rho_0}{\delta_{ij}}$. $\Psi_{1,i}(\eta_i(g))$ is strictly increasing with respect to $\eta_i(g)$ since for $x, y \in \mathbb{N}$, $x < y \Leftrightarrow \Psi_{1,i}(x) < \Psi_{1,i}(y)$. Hence the function $\pi_i(g, b)$ is convex. Since the inequality $\Psi_{2,ij}(x, b) - \Psi_{2,ij}(x-1, b) < 0$ holds for $\forall x \in \mathbb{N}$, π_i also satisfies the strategic substitutes property. □

The second term - namely the cost - is related with when the robot expects to do worse in regards to the network. According to the assumed communication strategy, as the number of its neighbors increases, the communication quality will deteriorate. This consideration is encoded by a cost function as:

$$\kappa_i(g, b) = c\eta_i(g) \tag{3.5}$$

where c is cost parameter [3] .

3.2.1. Pairwise Stability & Pairwise Nash Stability

As explained earlier, the payoff functions are not sufficient to determine what constitutes an acceptable assistance network for each robot. Again, pairwise stability is used to define acceptable assistance networks. The formal definition of pairwise stability is given by Definition 2.1 in Chapter 2. The existence of pairwise stable networks cannot be guaranteed in general. However, when payoff functions v_i of Equation 3.3 have local spillovers, convexity and strategic substitutes properties, the following result holds [64]:

Proposition 3.2. *If the payoff functions all have local spillovers, convexity and strategic substitutes properties, then there exists at least one pairwise stable network.*

Pairwise Nash stability is a refinement of pairwise stability based on Nash equilibrium [41] as given in Definition 2.2 in Chapter 2. With $\alpha-$submodularity[4] , the marginal gain associated with a group of edges already in the network is at least as

[3] Although not investigated in this work, the cost parameter c may become varying for each robot. Our stability and convergence results are also valid in case of the varying c.

[4] The definitions of both $\alpha-$submodularity and marginal gain are presented in Section 2.4.3.

high as the sum of individual edge marginal gains scaled by α. For the particular payoff functions, the marginal gains can be expressed as given in Lemma 3.3.

Lemma 3.3. *Let v_i, $i \in \mathcal{R}$ be a set of payoff functions defined as $v_i(g,b) = \pi_i(g,b) - c\eta_i(g)$. Then,*

$$\delta v_i(g,b,\ell) \;=\; |\ell|(1-c) + \sum_{ij\in\ell} \frac{\rho_0}{\eta_j(g)\delta_{ij}} \tag{3.6}$$

Proof. By definition, the marginal gain is equal to:

$$\begin{aligned} \delta v_i(g,b,\ell) &= v_i(g,b) - v_i(g-\ell,b) \\ &= \pi_i(g,b) - c\eta_i(g) - [\pi_i(g-\ell,b) - c\eta_i(g-\ell)] \\ &= \pi_i(g,b) - \pi_i(g-\ell,b) - c|\ell| \end{aligned} \tag{3.7}$$

Expanding the term $\pi_i(g,b) - \pi_i(g-\ell,b)$ in the right hand side of Equation 3.7

$$\begin{aligned} &= \eta_i(g) + \sum_{j\in\mathcal{N}(g)} \frac{\rho_0}{\eta_j(g)\delta_{ij}} - \left[\eta_i(g-\ell) + \sum_{j\in\mathcal{N}(g-\ell)} \frac{\rho_0}{\eta_j(g)\delta_{ij}}\right] \\ &= |\ell| + \sum_{ij\in\ell} \frac{\rho_0}{\eta_j(g)\delta_{ij}} \end{aligned} \tag{3.8}$$

Hence, the result. □

Theorem 2.8 provides the necessary and sufficient conditions on the payoff functions v_i, $i \in \mathcal{R}$, for the set of pairwise stable networks and the set of pairwise Nash stable networks to coincide. For the particular payoff functions, the conditions of Theorem 2.8 hold as shown in Proposition 3.4.

Proposition 3.4. *Consider payoff functions v_i, $i \in \mathcal{R}$ with $v_i(g,b) = \pi_i(g,b) - c\eta_i(g)$. All pairwise stable networks are also pairwise Nash stable.*

Proof. $\forall i \in \mathcal{R}$,$\forall g \in \mathcal{G}$, $\forall \ell \subseteq \mathcal{E}_i(g)$, consider $\delta v_i(g, b, \ell)$. By definition,

$$\begin{aligned} |\ell|(1-c) + \sum_{ij \in \ell} \frac{\rho_0}{\eta_j \delta_{ij}} &= \alpha \sum_{ij \in \ell} \left[1 - c + \frac{\rho_0}{\eta_j \delta_{ij}} \right] \\ &= |\ell|(1-c) + \sum_{ij \in \ell} \frac{\rho_0}{\eta_j \delta_{ij}} \end{aligned} \tag{3.9}$$

Now consider the left hand side of Equation 3.9. By Lemma 3.3

$$1 - c + \frac{\rho_0}{\eta_j(g) \delta_{ij}} = \delta v_i(g, b, ij)$$

This implies that

$$\delta v_i(g, b, \ell) = \sum_{ij \in \ell} \delta v_i(g, b, ij) \tag{3.10}$$

Hence $\alpha-$submodularity condition holds with $\alpha = 1$. The results follows by Theorem 2.8. □

3.3. Centralized Pairwise Games Based Networks

As acceptability is defined based on pairwise stability with respect to the payoff functions, the coordinator needs to employ a scheme that ensures the resulting topology to be pairwise stable. This is achieved via having the coordinator play centralized pairwise games utilizing the payoff functions [65]. If the payoff functions satisfy local spillovers, convexity and strategic substitutes properties, the resulting network is ensured of being pairwise stable - which implies that it is acceptable to all the robots in regards to their respective objectives.

3.3.1. Centralized Pairwise Games

Each CPG is a stochastic process that proceeds via a series of game moves until all the payoffs satisfy a termination condition. At the onset, all the robots relay their

state information b to the coordinator. Let $\tilde{g}_k(b)$ denote the network at the k th game move of the pairwise game with robot configuration state b. The b argument is omitted for notational simplicity whenever possible. In each game move, the coordinator does the following:

- Select a robot pair ij randomly,
- Determine the action to consider which is either to establish edge ij if $a_{ij} = 0$ or to sever it if otherwise.
- Compare v_i and v_j values with the payoffs resulting from this action and update the respective components of a as follows:

$$a_{ij} = \begin{cases} 0 & \text{if } v_i(\tilde{g}_k - ij, b) > v_i(\tilde{g}_k, b) \text{ or } v_j(\tilde{g}_k - ij, b) > v_j(\tilde{g}_k, b) \\ 1 & \text{if } v_i(\tilde{g}_k + ij, b) > v_i(\tilde{g}_k, b) \text{ and } v_j(\tilde{g}_k + ij, b) > v_j(\tilde{g}_k, b) \end{cases} \tag{3.11}$$

Thus, a game move between the pair ij either leads to

- The removal of an existing edge between them iff $v_i(\tilde{g}_k - ij, b) > v_i(\tilde{g}_k, b)$ or $v_j(\tilde{g}_k - ij, b) > v_j(\tilde{g}_k, b)$ or
- The setup of a new edge if $v_i(\tilde{g}_k + ij, b) > v_i(\tilde{g}_k, b)$ and $v_j(\tilde{g}_k + ij, b) > v_j(\tilde{g}_k, b)$.

The network state a changes throughout the sequence of game moves. All entries where there are no corresponding game moves being made remain unchanged. The algorithm in Figure 3.2 presents the centralized pairwise game of coordinator. The stochastic process continues until the network becomes pairwise stable. As this is a time-consuming check, it may be done only at sampled game moves depending on the network size as presented in the algorithm as shown in Figure 3.3. In each move, a pair is selected (line 5 of the algorithm shown in Figure 3.2). If these are connected in the network $\tilde{g}_k$, it considers breaking it up. Otherwise it considers the effect of forming an edge (lines 6-16 of the algorithm shown in Figure 3.2). Let $g^*(b)$ be the final network topology. Once the game terminates, the coordinator sends the network topology via transmitting $g^*(b)$ to the robots. The network topology does not change until the next

game.

Get information from all the robots
$\tilde{g}_0 \leftarrow g(t)$
$k \leftarrow 0$ ▷ Game move index
while $\tilde{g}_k$ is not pairwise stable **do**
    Select a pair ij randomly
    if $ij \in \mathcal{E}$ **then** ▷ Does edge ij exist?
        if $v_i(\tilde{g}_k - ij, b) > v_i(\tilde{g}_k, b)$ or $v_j(\tilde{g}_k - ij, b) > v_j(\tilde{g}_k, b)$ **then**
            $a_{ij} = 0$
            $\mathcal{E} \leftarrow \mathcal{E} \backslash ij$ ▷ Break up the edge ij
        end if
    else
        if $v_i(\tilde{g}_k + ij, b) > v_i(\tilde{g}_k, b)$ and $v_j(\tilde{g}_k + ij, b) > v_j(\tilde{g}_k, b)$ **then**
            $a_{ij} = 1$
            $\mathcal{E} \leftarrow \mathcal{E} \cup ij$ ▷ Establish the edge ij
        end if
    end if
    $k \leftarrow k + 1$
end while
$g^*(b) \leftarrow \tilde{g}_k$
Send $g^*(b)$ to robots

Figure 3.2. Centralized pairwise game at time t.

3.3.2. Convergence

Let $\mathcal{K}(b) = \{\tilde{g}_k(b)\}_{k=1}^{\infty}$ denote the sequence of networks generated throughout the CPG with initial state $\tilde{g}_0 = g(t)$. The sequence $\mathcal{K}(b)$ converges to a network $g^*(b)$ if $\lim_{k\to\infty} \tilde{g}_k(b) = g^*(b)$. Each CPG produces an improving path - namely a sequence of networks where the removal or the addition of the edge benefits the robot(s) whose consent is necessary for the change [38]. However, the improving path might lead to a cycle where a number of networks are repeatedly visited [38]. One way to ensure convergence to a pairwise stable network is to rule out the existence of cycles - as

1: **for all** $ij \in \mathcal{E}$ **do**
2: **if** $v_i(\tilde{g}_k - ij, b) > v_i(\tilde{g}_k, b)$ or $v_j(\tilde{g}_k - ij, b) > v_j(\tilde{g}_k, b)$ **then**
3: **return** $\tilde{g}_k$ is not pairwise stable
4: **end if**
5: **end for**
6: **for all** $ij \notin \mathcal{E}$ **do**
7: **if** $v_i(\tilde{g}_k + ij, b) > v_i(\tilde{g}_k, b)$ and $v_j(\tilde{g}_k + ij, b) > v_j(\tilde{g}_k, b)$ **then**
8: **return** $\tilde{g}_k$ is not pairwise stable
9: **end if**
10: **end for**
11: **return** $\tilde{g}_k$ is pairwise stable

Figure 3.3. Checking pairwise stability for network $\tilde{g}_k$.

presented in [66]:

Proposition 3.5. *If a pairwise stable network exists, there are no closed cycles.*

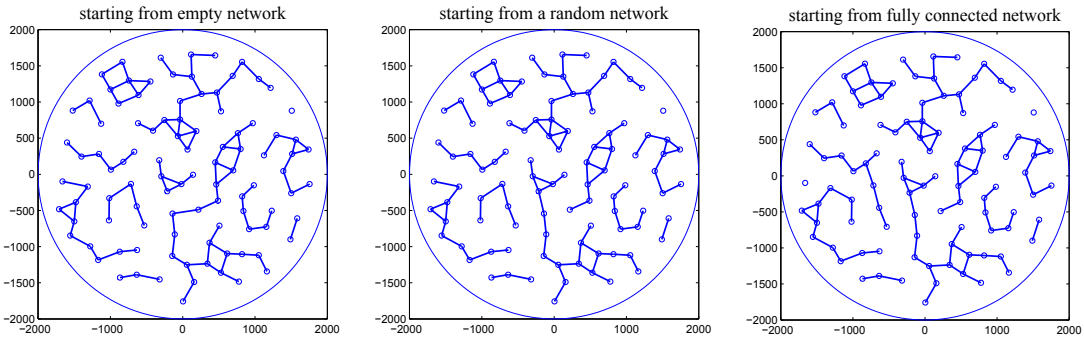
Note that Proposition 3.5 is a restatement of Lemma 1 of [38].

Corollary 3.6. *If a pairwise stable network exists, starting from any network, a CPG converges to a pairwise stable network with probability 1.*

The payoff functions ensure the existence of pairwise stable network and cycles are ruled out by Proposition 3.5, a CGP converges to a pairwise stable network with probability 1.

3.4. Simulations

In our simulations, robot teams with populations of 50 and 100 in a workspace of radius 2 km are considered. The robots are cylinder-shaped robots with radii 25 cm. Pairwise stable networks for a random robot configuration state b with three different initial network topologies $\tilde{g}_0$ are as shown in Figure 3.4. The link cost parameter of



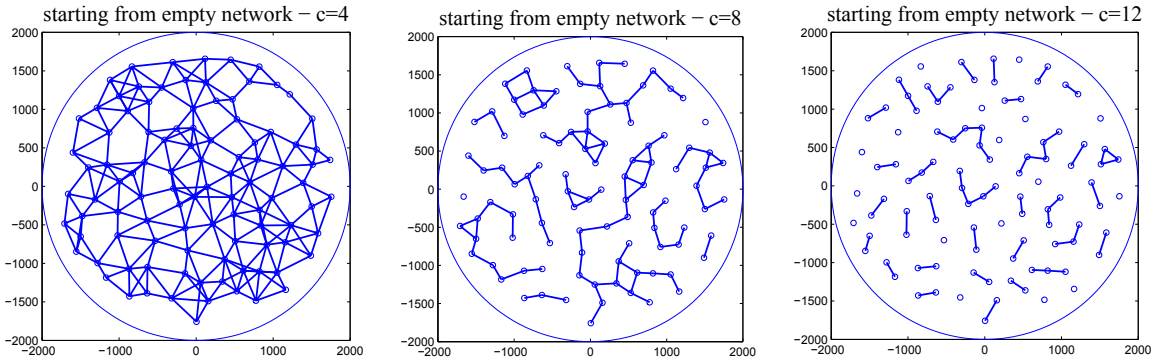
(a) (b) (c)

Figure 3.4. Pairwise stable networks with 100 robots. (a) With initial network $\tilde{g}_0(b) = \emptyset$; (b) With random initial network $\tilde{g}_0(b) = g'$; (c) With initial network fully connected $\tilde{g}_0(b) = g^r$.

payoff functions is $c = 8$ so that point-to-point communications within around 300 meters become viable. It is observed that the resulting networks - while being similar - have nevertheless some differences among them. This suggests that pairwise stable networks are not unique.

3.4.1. Statistical Analysis

In the next set of simulations, the effect of cost parameter c on the resulting pairwise stable networks is investigated. The robot configuration state b is as previously taken while the initial network topology is $\tilde{g}_0(b) = \emptyset$. The link cost parameter c is varied between 4 to 12. Sample pairwise stable networks are as shown in Figure 3.5. As the link cost decreases, the resulting network becomes connected while connectivity also increases. On the other hand, if the link cost is increased, pairwise networks divide the robots into a number of distinct groups where this number is controlled by the cost parameter and where each group is minimally connected. With $c = 8$, the network has less than four connections per robot. In order to study real-time performance, processing time is studied. A mobile robot in our laboratory with Intel Atom 1.6 GHz processor and 2 GB RAM is programmed as the communication coordinator. For 10 random robot configuration states and for three different initial network topologies, the



(a) (b) (c)

Figure 3.5. Pairwise stable networks with 100 robots for varying cost parameters with empty initial network. (a) $c = 4$; (b) $c = 8$; (c) $c = 12$.

coordinator run pairwise games 10 times and generates the pairwise stable networks. This is repeated for two different robot team sizes. Pairwise stability is checked every 500 game moves for 50 robots and every 1000 game moves for 100 robots – in order to expedite convergence time as discussed previously. The resulting number of game moves and the processing times are presented in Table 3.1.

Next, the number of game moves necessary for attaining pairwise stability is investigated. Of course, the cardinality of $\mathcal{A}$ – namely $|\mathcal{A}| = 2^{\binom{r}{2}}$ – is extremely large for robot team sizes $r = 50, 100$. The number of game moves gives the number of different states visited in $\mathcal{A}$ until a pairwise stable network is reached. In this perspective, it is observed that on the average, 3000-4500 game moves are made with 50 robots. This number goes up to 14000-20000 game moves with 100 robots. Timewise, this corresponds to 0.05-0.2 seconds for 50 robots and 1-3 seconds for 100 robots. Let it be remarked that in a real multirobot application, as the change in the robots' configuration states will be gradual between consecutive pairwise games, there will not be as much difference between the consecutive initial network topologies. Hence, both the number of game moves and the processing times will be expected to be considerably lower.

Table 3.1. Number of game moves & Processing time.

		Num. of game moves (x1000) – Time (sec.)			
$\boldsymbol{r}$	$\tilde{\boldsymbol{g}}_0(\boldsymbol{b})$	**Min.**	**Max.**	**Mean**	**Std. Dev.**
50	$\emptyset$	2 – 0.04	4 – 0.09	3.05 – 0.07	0.5 – 0.015
	g'	3.5 – 0.14	5 – 0.2	4.31 – 0.17	0.43 – 0.013
	g^r	3.5 – 0.14	5.5 – 0.21	4.25 – 0.17	0.46 – 0.016
100	$\emptyset$	10 – 0.71	17 – 1.42	14.44 – 1.15	1.06 – 0.1
	g'	17 – 2.4	22 – 2.8	19.4 – 2.62	0.88 – 0.07
	g^r	17 – 2.32	22 – 2.82	19.26 – 2.6	1.12 – 0.11

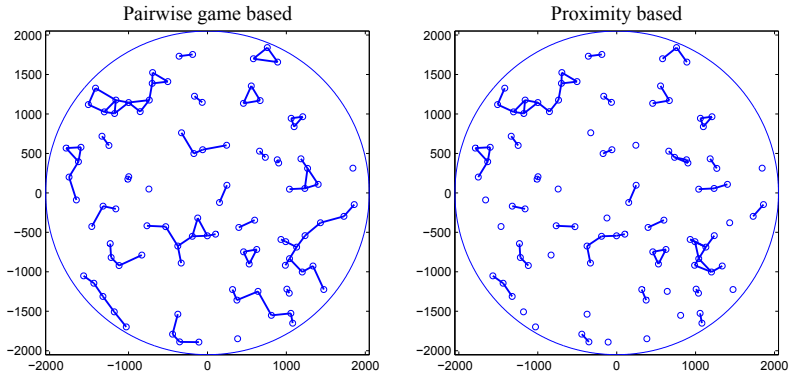
3.4.2. Comparative Analysis

In this Section, the proposed approach is compared with the proximity based communication where robots establish links with all robots within a fixed communication range ρ_c meters. Of course, while proximity-based networks are essentially randomly generated networks, pairwise stable networks are generated with a specific set of objectives in mind. In order to make the comparison as fair as possible, ρ_c is selected such that average communication payoff value $\bar{v}_i(g, b)$ is maximized:

$$\rho_c = \arg \max_{\rho_c \in [100,500]} \bar{v}_i(g, b) \tag{3.12}$$

The resulting $\rho_c = 240$ meters is used in the proximity based approach.

The simulations are done with a team of 100 robots with link cost $c = 8$ at 100 random configuration states b with the network topology initialized to $g_0 = \emptyset$. Sample network topologies for the proposed game based and standard proximity based approaches are shown in Figure 3.6. The resulting networks are compared with respect to four measures: (i) The minimum value of individual payoff functions v_i, (ii) The average of the ratio of the number of missing links over the number of missing and



(a) (b)

Figure 3.6. Network topologies: (a) Centralized pairwise game based, (b) Proximity based.

established links, (iii) The average of the ratio of the number of established, but not desired links over the number of total established links, and (iv) The percentage of networks that are pairwise stable. The results are as presented in Table 3.2. It is observed that while in centralized pairwise game approach, none of the robots have negative communication payoff values, this is not the case for proximity based approach. A 34.9% result in the average ratio of the number of missing links over the number of non-established links corresponds to an average of 37 missing links in 100 robot scenario. This means that robots are missing the benefits they would reap if they had actually established these links. The third measure is around 5.8% that indicates that the networks have an average of four links that would be better off if not established. Finally as expected, none of the networks are pairwise stable - even in case when the proximity parameter ρ_c is optimally selected with respect to the average payoff values.

3.5. Conclusion

This chapter studies the problem of effective communication network topologies. It proposes a centralized model of how the network may evolve over time. In this model, a network coordinator updates the network topology. Periodically, the robots contact the network coordinator. In turn, the communication coordinator considers the individual communication payoff functions of all the robots, their current states

Table 3.2. Centralized pairwise game based vs. proximity based.

	Pairwise Game	**Proximity**
Minimum v_i	0.15	-0.12
Missing links %	0	34.9
Undesired links %	0	5.8
Pairwise stable %	100	0

and the current network simultaneously and finds a network topology acceptable to all the robots. This is accomplished via pairwise games which are basically iterated processes consisting of a sequence of game moves. In case the communication payoff functions satisfy local spillovers, convexity and strategic substitutes properties, each pairwise game is ensured of convergence to a pairwise stable network. Simulation results provide insights on the resulting network topology and the processing time as well as comparative results with the proximity based approach.

4. ASSISTANCE NETWORKS

In multi-robot systems operating in unknown environments, robots need to handle tasks they encounter dynamically [67,68]. If these tasks require cooperative actions, then the robots can complete them only with assistance from other robots [69]. For example, such assistance may comprise of sharing of information, providing additional resources or giving physical help [70–72]. While these tasks and their particular requirements will vary depending on the application, in all, the question of which robot should assist which robot needs to be addressed effectively in order for the tasks to be successfully completed. This is a challenging task. First of all, as tasks are dynamic, they occur at unpredictable places or times which implies that the assisting robots cannot be assigned a priori. Secondly, the problem is exacerbated by the fact that some robots will be busy when others are seeking assistance. Finally, there will be task related constraints that need to be taken into account.

In this chapter, the problem of dynamic and cooperative tasks in multi-robot systems is considered. In particular, all the robots are assumed to be capable of handling any of the encountered tasks, but only with the assistance of another robot[5] . Robots are assumed to give assistance when required - except when they are actually engaged in a task themselves. Finally, since robots can be engaged in one task at a time, a task with timeout cannot be started within a fixed period. As posed, this is an instance of dynamic multirobot task allocation (MRTA) problem [68, 73]. In the taxonomy of MRTA problems consisting of three categories (single-task (ST) vs multitask (MT) robots, single-robot (SR) vs multirobot (MR) tasks and instantaneous (IA) vs time-extended (TA) assignment), it falls in the category of ST-MR-IA problems. It is ST since each robot is capable of handling only one task at a time It is MR as each robot needs assistance from another robot. Finally, it is IA since robots cannot plan for future tasks as they are completely unknown a priori. In a more recent taxonomy of MRTA problems, additional dimensions associated with interrelated utilities and

[5]While only tasks that can be completed with assistance from one robot are considered, this approach can be extended to multi-robot assistance problems.

constraints are introduced [68]. In this new taxonomy, the problem falls in the domain of MRTA problems with cross-schedule dependencies where schedule optimization is required in order to enable coordination among robots. As discussed in the sequel, most previous work aim to find an explicit assignment of assisting robots. As this has proven to be strongly NP-hard, approximate solutions with an emphasis on computational feasibility and practical applicability need to be developed [68, 74].

The contribution of this chapter is to propose a novel approach to this problem. In particular, the concept of 'assistance networks' is introduced. In contrast to finding an explicit assignment of assisting robots, which is a challenging problem as explained, the assistance network designates potential helpmates in an implicit manner based on its topology. Each robot - when faced with a task - can seek an assisting robot among its immediate neighbors in the network in a decentralized manner. As time evolves and robots encounter new tasks, the topology of the assistance network needs to be updated. This is done by a network coordinator in a centralized manner. In order for the network topology to be acceptable to all the robots, their assistance related preferences are taken into account. The update process is modeled as a centralized pairwise game (CPG). With certain restrictions on robots' preferences, the CPG is shown to converge to a pairwise stable network [65]. The assistance network is ensured of having an acceptable topology where robots flexibly get assistance or give assistance during their operation. Simulations and experimental results demonstrate that the assistance network enables effective allocation of assisting robots while being easily implemented with a multi-robot system.

The outline of the Chapter is as follows: First, related work that address the ST-MR-IA problems is reviewed in Section 4.1. Dynamic and cooperative tasks are described in Section 4.2. The assistance networks are introduced in Section 4.4 where the role of the assistance network coordinator with respect to the assistance network is also explained and the payoff functions are defined. In Section 4.6, extensive simulation results demonstrate that the robots are capable of completing tasks effectively in a variety of different scenarios - including a comparative study with proximity-based assistance. Experimental results with a team of five mobile robots as presented in

Section 4.7 verify the practicality of assistance networks for dynamic and cooperative tasks. The Chapter concludes with a general discussion and future directions.

4.1. Related Literature

The ST-MR-IA problem is most commonly viewed as an instance of optimal assignment of a set of tasks to the robots based on maximizing the overall performance while taking their individual performances into consideration [73, 75]. The proposed approaches differ primarily in the formulation of the constrained optimization problem and how it is solved [76]. While the definition of optimization problems vary from finding optimal coalition structures to utility maximization, optimization methods can be categorized as being centralized or distributed [77]. Centralized approaches are simpler to implement, but become practically infeasible as the number of tasks and robots increase. On the other hand, while distributed approaches are advantageous with respect to reliability and scalability, the efficiency of the resulting solutions are harder to ensure. In practice, systems may not conform to a strict centralized/decentralized dichotomy and may contain both elements [78].

Most commonly, the optimization problem is defined as finding the optimal partitioning of the robots into task-specific coalitions - motivated by game-theoretic work that aims to generate optimal coalition structures [79]. The set partitioning problem has been shown to be strongly hard where an exponential number of coalition structures needs to be searched [79]. It is difficult to approximate the problem using techniques from combinatorial auctions [80]. For robotics applications, the tasks have been put in a taxonomy based on demands, resources and profit objectives [81]. While solutions for five distinct classes of the problem are proposed, these solutions may not be applicable in robotic scenarios where the capabilities on the robots are sharable between different coalitions [82]. In this perspective, many heuristic algorithms have been developed [73]. An anytime algorithm is shown to have bounded solution with a minimal search [83]. Improved solutions are obtained such as a polynomial time dynamic programming approach in the case of agents of different types being indistinguishable or solutions to be within a constant factor of the optimal utility the population of coali-

tions are restricted [84]. In case of constrained resources, a greedy optimal solution is proposed via a leader follower coalition method where coalition utility is maximized for every assigned task [85]. Two natural greedy heuristics are extended via a new greedy heuristic that considers the expected loss of utility due to the assigned robots and task as an offset and uses the offset utility for task assignment [74].

Distributed approaches solve the constrained optimization problem in a decentralized manner [86]. In a class of problems known as distributed constraint optimization problems (DCOPs), each robot or group controls one set of variables and together they have the joint goal of maximizing a global objective function [44, 87]. Complete algorithms maximize a global objective function, however are not feasible in ST-MR-IA problems due to the inherent complexity of the problem. In best response algorithms, each robot or group reacts on the basis of local knowledge so that there is no need for tree-like communication [88]. However, since each group's utility is defined as the sum of the terms of the global objective function, each decision-making has to consider all the other payoffs as well [44]. In social networks, the robots are distributed over a given network and only neighboring agents are allowed to work together in a task [89,90]. As the complexity of the problem remains NP-complete and it is not approximable within some factor, distributed efficient greedy algorithms require that robots have only local knowledge about tasks and resources. However, in these work, the network topology is assumed to be given a priori and fixed. As such, the robots face with the difficult problem of searching a factorial sized space for an optimal strategy which naturally has adverse effects on performance [91].

Alternatively, the desired allocation could consider the preferences of all the robots individually. Here, in contrast to optimizing with respect to a single objective, a set of objective functions is simultaneously optimized. In general, multiobjective optimization may lead to different solutions as compared to optimizing with respect to a single objective [44]. The robots choose these tasks via considering individual motivations in a fully distributed, behavior-based architecture [92]. In market-based approaches, the robots submit bids based on their respective expected costs and the robot with the best bid takes the task [93–95]. The auctions can be conducted either

in centralized or distributed manner. The centralized auctioneer provides a framework for considering optimization of a global objective and can be combinatorial [96] or greedy [97]. However, it has the communication related and computational disadvantages of the fully centralized approaches [77]. Distributed auction approaches are derived from centralized auctions using regional opportunistic centralization [93, 98–100]. An approach based on token passing avoids the large communication bandwidth requirement from which market-based approaches suffer [101]. In all, the quality of the solutions produced needs careful analysis - with the growth of robot team size [67].

Multiobjective decision-making has been studied extensively in game theory [39, 45]. Here, each player is associated with its individual considerations and the goal is to find a solution that is acceptable to all the players simultaneously. This is in general different from optimizing with respect to a single global objective function if even all the robots have the same objectives. It is known that game-theoretic solutions and Pareto-optimal solutions (namely the solution to a single objective function) may differ when the individual objective functions are dependent on those of other robots [102]. The objective functions are not sufficient to determine what is an acceptable solution for each robot - as they will be coupled with each other. In general, it will not be possible to find a single solution that simultaneously optimizes each objective. Additional preference information regarding what constitutes an acceptable solution is introduced. The simplest definitions are pairwise stability and pairwise Nash stability [39]. A problem similar to ours is modeled as a Markov game [91]. However, due to computational intractability, the acceptability of the robot schedules – that are explicitly generated via a sequence of approximating potential games – cannot be guaranteed.

In this chapter, game-theoretic ideas are used from a different perspective. In particular, the concept of assistance networks is introduced for our particular case of ST-MR-IA problem. As robots encounter new tasks, there is no explicit assignment of assisting robots to them. Rather, which robot can help which other robot is loosely specified via the assistance network - similar to social networks as discussed previously. A model of network formation needs to specify how the robots establish edges together with a network equilibrium concept. Here, the model that has been developed for

centralized network topology as explained in Chapter 3 is adopted.

4.2. Dynamic and Cooperative Tasks

As a robot is moving around in its workspace, it will come across a number of tasks. All robots are assumed to be capable of handling encountered tasks when assisted by another robot. There is no a priori information regarding their spatial locations or when a robot will encounter one. One way to give the robot an ability to respond to encountered tasks is to give it an internal state where it can store its knowledge of the tasks. Let $h_k^i = \left[h_{k,1}^i \, h_{k,2}^i\right]$ denote the k th task encountered by robot i. It is associated with two properties - its time of encounter as denoted by $h_{k,1}^i \in \mathbb{R}$ and its status $h_{k,2}^i$ that has discrete set of values defined as:

$$h_{k,2}^i \;=\; \begin{cases} 1 & \text{if task } k \text{ was completed} \\ 1^- & \text{if task } k \text{ is being handled} \\ 0 & \text{if task } k \text{ is waiting to be handled} \\ -1 & \text{if task } k \text{ could not be completed} \end{cases}$$

When a robot encounters a task, it adds it to the set of tasks $\mathcal{H}^i$ as $\mathcal{H}^i = \left\{h_1^i, h_2^i, \ldots h_{n_i}^i\right\}$. where n_i is number of encountered tasks. The value n_i is initially zero and increases over time. The status of a task is initialized as waiting $h_{k,2}^i = 0$. It is assumed that tasks encountered by each robot are particular to it. Each task can be completed only with assistance from one other robot.

4.3. Task Automaton

The robots handle the encountered tasks in a decentralized manner based on the current assistance network. The task automaton is associated with four states: 'idle', 'handling', 'assisting' and 'waiting' as shown in Figure 4.1. Normally, the robot is in idle state. If the robot is handling a task or assisting another robot, it is in the handling or assisting states respectively. If it is waiting for assistance, it is in the waiting state.

The rules of this process are as follows:

(i) A robot i can either handle one task or assist another one robot at a time.

(ii) When in idle state, it considers a task in $\mathcal{H}^i$ and requests assistance from immediate neighbors $\mathcal{N}_i(g)$ as designated by the assistance network depending on availability and proximity.

(iii) In case its assistance request is accepted, the robot goes into handling state and stays in this state for duration of Δt_h.

(iv) A robot accepts an assistance request if it is in the idle state. Once it starts assisting, it remains in the assisting state for duration of Δt_h regardless of any network update.

(v) A robot needs to handle each task within a maximum period Δt_o - after which a timeout occurs.

(vi) In case a robot cannot get any assistance, it goes into the waiting state and remains in this state until either it gets assistance or timeout occurs.

From the robots' perspective, the first rule states that each robot cannot simultaneously handle a task and assist another robot. If the robots are capable of assisting other robots while handling their own tasks, this rule may be relaxed accordingly. The second rule implies that potential helpmates are implicitly specified by their local neighborhood $\mathcal{N}_i(g)$ as designated by assistance network g. The third and fourth rules constrain the resources of the robot pair engaged in a task for a duration of Δt_h. Note handling times and assistance times are taken to be same - referred to as task time in the sequel. The fifth rule requires the robots need to start handling tasks within a certain amount of time. Otherwise, time-outs occur which imply failure to complete the corresponding tasks. Finally, the last rule implies that a robot cannot handle a task unless it gets assistance. In case of a timeout, the robot fails to handle the particular task and it goes into idle state. If the tasks are not homogeneous, the parameters Δt_h and Δt_o may vary depending on the type of the tasks. However, for the sake of simplicity, in this work, they are assumed to be the same for all the tasks. The robot updates its tasks' states according to this automaton. The robots need to handle as many tasks as they can. Let it be noted that the robots do not check if a task they

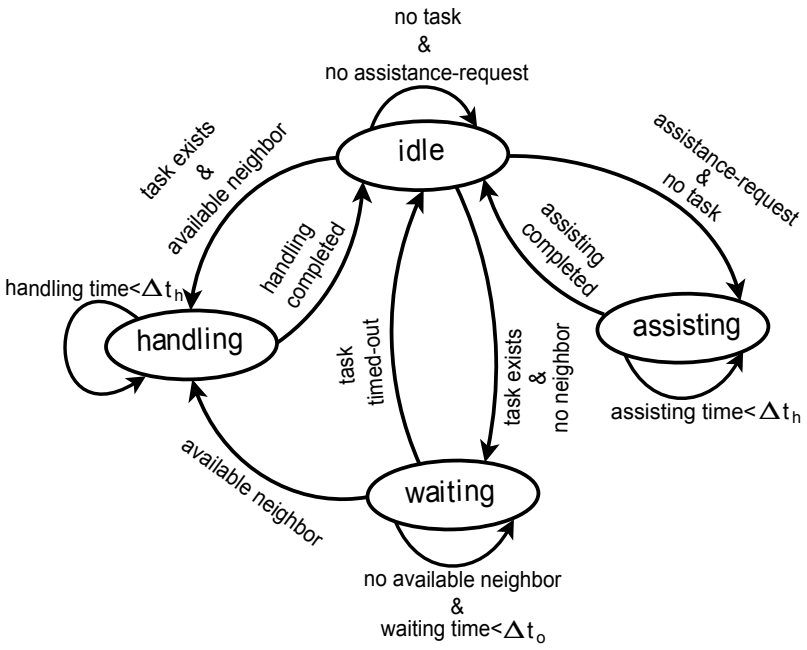


Figure 4.1. Robot's task automaton.

have encountered has been previously handled by another robot. As such, a task may possibly be handled by different robots. This can be avoided via either marking the handled task or global communication and coordination - which may not be desired or possible. Furthermore, it may be advantageous to handle tasks multiple times in applications where additional information regarding tasks over time is preferable.

4.4. Assistance Network

The assistance network enables the robots to choose their helpmates in a decentralized manner - namely the robots seek assistance from neighboring robots in the assistance network. It is defined as $g(t) \in \mathcal{G}$ where $\mathcal{G} = \{g' | g' \subseteq g^r\}$ is the set of all possible graphs on $\mathcal{R}$ and g^r is the complete graph. The set of edges $\mathcal{E}(g) \subseteq Q$ represents the robot pairs i and j between which a direct cooperation edge is established. Note that the case when $g(t) = g^r$ corresponds to complete cooperation in which all the robots can assist with each other. The 1-hop neighborhoods $\mathcal{N}_i(g) \triangleq \{j \in \mathcal{R} \,|\, ij \in \mathcal{E}\}$ correspond to the edges of graph g. The cardinality of $\mathcal{N}_i(g)$ is denoted by $\eta_i(g)$. It is assumed that the robots are able to get assistance with the robots in $\mathcal{N}_i(g)$ using

point-to-point communication.

The assistance network topology can vary from being fully connected ($\eta_i(g) = r-1, \forall i \in \mathcal{R}$) to completely disconnected ($\eta_i(g) = 0$). In the former case, the helpmate selection process becomes critical as all robots can potentially request assistance from all other robots and finding a robot that is free to help without time-outs faces serious scalability challenges with the growth of robot team size [2, 67]. In the latter case, the robots will have no potential helpmates. Hence, both cases are unacceptable.

4.5. Assistance Network Coordinator

A network coordinator is responsible for determining the assistance network topology. It finds a topology that is acceptable to all the robots via centralized pairwise games as presented in Section 3.3 of Chapter 3. Again, acceptability is defined based on payoff functions encoding the robots' preferences individually as $v_i : \mathcal{G} \times \mathcal{F} \rightarrow \mathbb{R}$. While distance is one criterion in determining the admissible robots, it is not the sole criterion – as in case of nearby robots being occupied themselves, time-outs are likely to occur. Furthermore, if there are many robots that are close to each other, the number of potential assisting robots may not be large. Hence, there are other considerations in designating the assisting robots. For example, robots that likely to assist fewer robots will be preferred. In this perspective, each robot's preferences are related with the gain and cost associated to having a particular neighborhood $\mathcal{N}_i(g)$. Correspondingly, the payoff functions $v_i : \mathcal{G} \times \mathcal{F} \rightarrow \mathbb{R}$ are comprised of two terms encoding gain and cost respectively as in Equation 3.3.

The gain $\pi_i(g, b)$ is related with when the robot expects to do better in regards to the assistance network. Common sense dictates that it should be able to get assistance whenever it requests to do so. Hence, waiting time for help should be minimal. For this, with no other considerations, each robot would prefer to have edges with as many robots as possible as this certainly increases its chances of getting a help. However, having as many neighbors as possible does not ensure immediate help as response time from each will deteriorate as the number of their respective neighbors increases. Thus,

at the same time, these neighbors should be ready to assist themselves which implies that they should be linked with as few robots as possible. These considerations are encoded by a gain function defined as:

$$\begin{aligned} \Psi_{1,i}(\eta_i(g)) &= \eta_i(g) \\ \Psi_{2,ij}(\eta_j(g), b) &= \frac{1}{\eta_j(g)} \frac{\rho_0}{\delta_{ij}} \end{aligned} \tag{4.1}$$

where for each pair ij of robots, $\delta_{ij} = \| b_i - b_j \|$ denotes the robots' pairwise distance. The term $\Psi_{1,i}$ increases in the number of its collaborative edges and $\Psi_{2,ij}$ decreases in the number of its neighbors' edges. The term $\Psi_{2,ij}$ decreases in the average distance between its neighbors. Together, they imply that each robot prefers having as many close neighbors as possible that in turn have few neighbors themselves - as this would increase the likelihood of getting assistance in shorter time.

The cost term is related with when the robot expects to do worse in regards to the assistance network. As the number of its neighbors increases, its decision-making requires more of its resources and hence slows it down. This consideration is encoded by a cost function as:

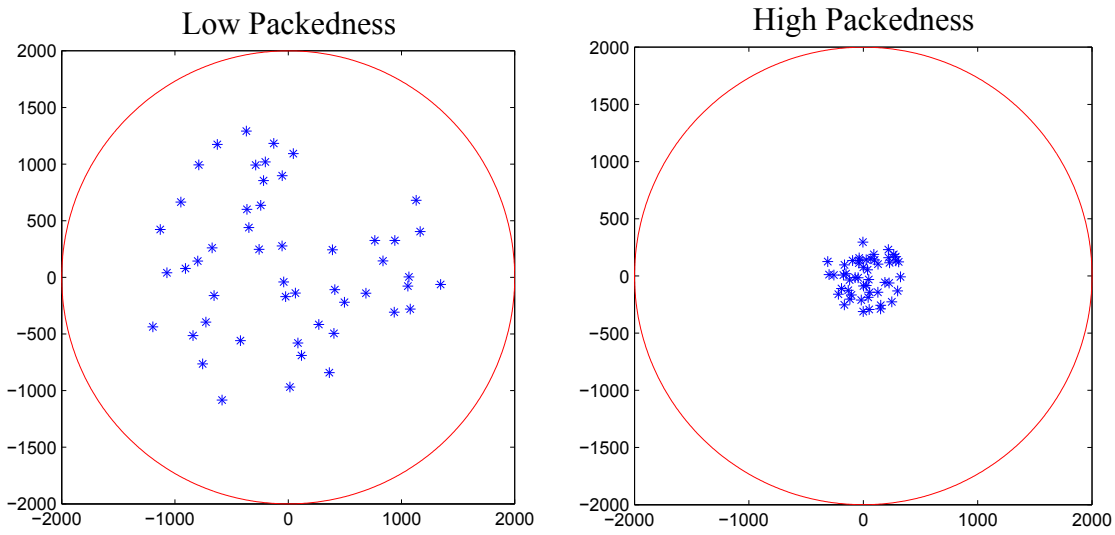
$$\kappa_i(g, b) = c\eta_i(g) \tag{4.2}$$

where c is cost parameter. It is observed that the payoff functions have local spillovers, convexity and strategic substitutes properties [63, 64]. As such, the stability and convergence results presented in Section 3.3 of Chapter 3 prevail.

4.6. Simulation Results

4.6.1. CPG-Based Assistance Networks

First task handling with CPG-based assistance networks is considered. For this purpose, extensive simulations have been conducted with 50 robots placed in a

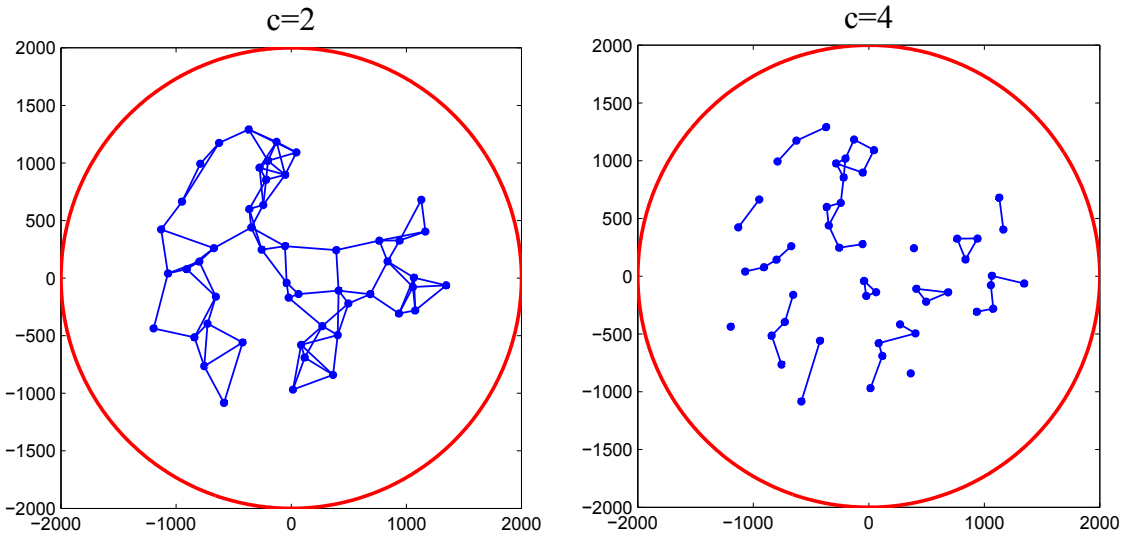


(a) (b)

Figure 4.2. Robot missions in a 12.5 km^2 workspace. (a) Low packed mission; (b) High packed mission.

workspace of radius 2 km. The robots are in a patrolling mission. As robots encounter tasks, they need to take cooperative action. In the simulations, two levels of robots' packedness (low or high) are considered as shown in Figure 4.2. In missions with low packedness, the robots are relatively distant from each other when they encounter tasks. In contrast, with high packedness, the robots are located much closer to each other. Depending on the level of packedness, the mission duration t_f varies from 140 minutes to 196 minutes. The assistance networks are formed with two levels of cost parameter $c \in \{2, 4\}$. Initially, the robots have no potential assisting robots - namely $\mathcal{E}(g(0)) = \emptyset$. Note that the cardinality of $\mathcal{A}$ – namely $|\mathcal{A}| = 2^{\binom{r}{2}}$ – is extremely large even for robot team size $r = 50$. The number of game moves gives the number of different states visited in $\mathcal{A}$ until a pairwise stable network is reached. The convergence times of pairwise games have been investigated [65]. For the worst case scenarios of 50 robots, an average of 3000-4500 game moves are required before reaching pairwise stability. However if the changes in the robots' states are gradual with respect to network update period T_g, then consecutive network topologies will not differ much. In our case, the network update period $T_g = 2.5$ minutes. Hence, the number of game moves is expected to be considerably lower. In our simulations, the pairwise stability

is also checked every 500 game moves. Timewise, this corresponds to 0.05-0.2 seconds for 50 robots. Sample pairwise stable networks are as shown in Figure 4.3. Increasing c divides robots into distinct components - which implies that the parameter c controls the assistance network density. For each level of packedness and c value, 20 different missions are conducted where robots all start at random initial locations. Hence, 80 different missions are generated.

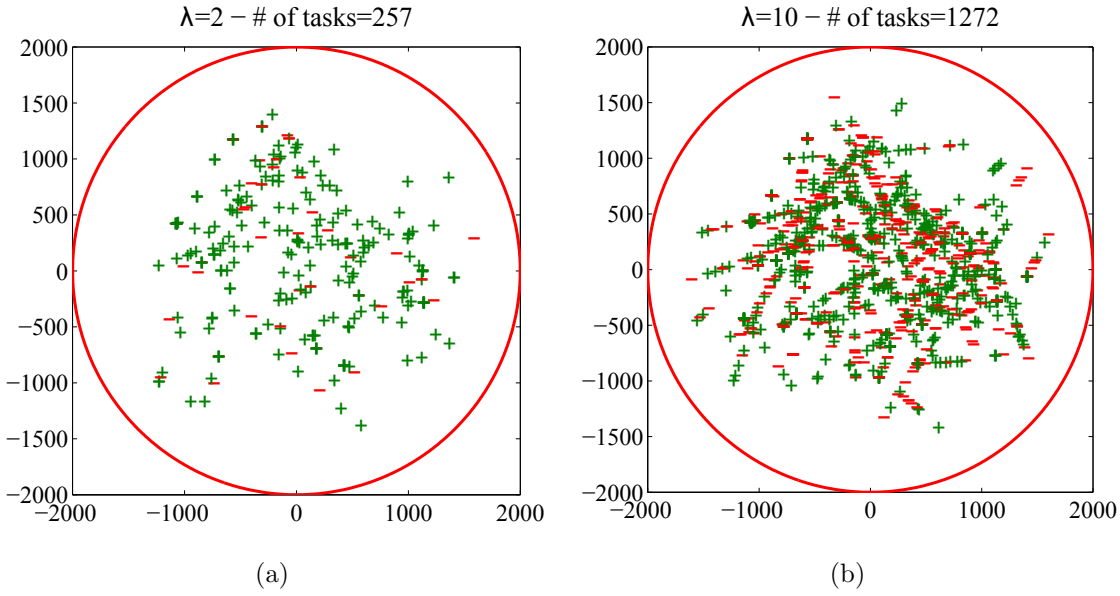


(a) (b)

Figure 4.3. Sample pairwise stable assistance networks in missions with low packedness for varying cost parameter c. (a) $c = 2$, (b) $c = 4$.

Next, tasks are considered. As the focus is on the effective allocation of assisting robots so that these tasks can be completed, tasks encountered by each robot are generated as a Poisson process with parameter $\lambda \in \{2, 10\}$. As the average number of tasks in a time interval δt is $\lambda \delta t$, the number of tasks will increase with increased λ value – hence making the mission more difficult and challenging. With $\lambda = 2$, the total number of tasks vary between 250 and 350 which corresponds to an average of 5-6 tasks per robot. The total number of tasks goes up to 1200 - 1650 with $\lambda = 10$. In this case, each robot encounters on the average 29 - 33 tasks. Sample spatial distributions of tasks encountered in a mission for different λ values are as shown in Figure 4.4 where there are 257 and 1272 tasks respectively. For each λ value, 10 different task scenarios are generated. The task duration varies from short to long as $\Delta t_h \in \{30, 300\}$ seconds.

Similarly time-out parameter is varied as $\Delta t_o \in \{60, 180\}$ seconds. Hence, there are 80 different task scenarios.



(a) (b)

Figure 4.4. Sample missions with low packedness. Robots encounter 257 tasks with $\lambda = 2$ while this number goes up to 1272 tasks with $\lambda = 10$. Note that these tasks are not known a priori and can be completed only with assistance of another robot. With $c = 2$, $\Delta t_h = 300$ and $\Delta t_o = 60$, the completed tasks are as shown by green plus signs while timed out tasks are shown by red minus signs.

Table 4.1. Simulation results for varying cost c in CPG-based assistance.

λ	c	2				4			
	Δt_h sec	30		300		30		300	
	Δt_o sec	60	180	60	180	60	180	60	180
2	$\bar{M}_1\%$	10.8				4.5			
	$\bar{M}_2\%$	0.2	0.1	12.9	6.9	1.5	1.3	16.1	9.4
	$\bar{M}_3\%$	7.2	7.2	8.2	8.4	7.1	7.2	7.8	7.9
10	$\bar{M}_1\%$	10.8				4.5			
	$\bar{M}_2\%$	0.2	0.1	44.5	36.1	4.1	3.6	53.6	47.3
	$\bar{M}_3\%$	7.8	7.8	10.4	11.2	7.2	7.3	8.0	8.2

Table 4.2. Simulation results for varying packedness in CPG-based assistance.

λ	Packedness	Low				High			
	Δt_h sec	30		300		30		300	
	Δt_o sec	60	180	60	180	60	180	60	180
2	$\bar{M}_1\%$	5.5				9.5			
	$\bar{M}_2\%$	1.6	1.3	16.6	9.7	0.9	0.8	14.2	8.1
	$\bar{M}_3\%$	9.2	9.3	10.1	10.3	5.7	5.8	6.3	6.5
10	$\bar{M}_1\%$	5.5				9.5			
	$\bar{M}_2\%$	1.7	1.4	50.5	43.6	1.0	0.8	46.0	37.9
	$\bar{M}_3\%$	9.8	9.9	11.7	12.1	6.1	6.1	7.6	8.0

Then, the multi-robot system is simulated with 80 different missions and 80 different task scenarios - leading to 6400 mission-task scenarios. A sample mission-task scenario with low packedness is as shown in Figure 4.4. where the assistance network is formed with cost parameter $c = 2$. The task automaton is associated with parameters $\Delta t_h = 5$ minutes and $\Delta t_o = 60$ seconds. Tasks that are successfully completed are shown in green while those that were timed-out are in red. When the number of tasks are relatively fewer ($\lambda = 2$), 84% of the 257 tasks are completed. As the number of tasks is increased ($\lambda = 10$), the percentage of the completed tasks goes down to 54% with 687 out of the 1272 tasks being completed.

Statistical results are as shown in Table 4.1 and Table 4.2 - where the variations are with respect to the cost parameter c or packedness. The results are analyzed with respect to three performance measures:

- M_1 - Mission network density: It is defined as:

$$M_1 = 100 \frac{1}{N_g r(r-1)} \sum_{l=1}^{N_g} \sum_{i \in \mathcal{R}} \eta_i(g(lT_g)) \tag{4.3}$$

where $N_g = \frac{t_f}{T_g}$ is the number of times the assistance network is updated in a mission with completion time t_f. Of course, t_f will vary from mission to mission which implies that N_g will vary accordingly. If $M_1 = 100$, the network is maximal with the number of edges $\frac{r(r-1)}{2}$. In this case, each robot has to find a particular assisting robot from a larger set - which implies higher computational and communication requirements. With $M_1 < 100$, the number of edges goes down to M_1 percent of $\frac{r(r-1)}{2}$. Hence, it is preferable to have M_1 as small as possible.

- M_2 - Percentage of timed-out tasks: The goal is to have a low M_2. Depending on the occurrence of tasks, task duration Δt_h and time-out duration Δt_o, it may not be possible to have $M_2 = 0$. In this case, completing only a percentage of tasks is unavoidable.
- M_3 - Normalized pairwise distance: The average distance between all the caller-assisting robot pairs during the course of a mission normalized by the workspace diameter $2\rho_o$. With higher value, the computational requirements associated with the assisting robot selection and assistance are both expected to increase. Thus, M_3 should be as low as possible.

In the simulations, average values of these measures - denoted as $\bar{M}_1$, $\bar{M}_2$ and $\bar{M}_3$ respectively - are computed. If there are too many tasks, it may not be possible to have $\bar{M}_2 = 0$. An optimistic upper bound on the maximum number of tasks that can be completed can be computed - assuming a topology with disconnected robot pairs. Thus, each robot has only one neighbor. Furthermore, suppose that the tasks occur in an alternating manner for each robot which implies that each robot can complete one task every $2\Delta t_h$ periods. Thus, in a r robot team with mission of duration t_f, the maximal number of tasks that can be completed without time-outs will be bounded above by $N_H = r\frac{t_f}{\Delta t_h}$. In our case, these roughly correspond to 16500 tasks for $\Delta t_h = 30$ and 1650 tasks for $\Delta t_h = 300$. Of course, as this is an optimistic upper bound, the number of tasks that can be completed is expected to be comparatively smaller.

The network density $\bar{M}_1$ is the highest with cost $c = 2$ - which implies that the robots have the opportunity to get assistance from more robots as compared to a higher cost value. The percentage of timed-out tasks $\bar{M}_2$ varies between 0.1% to 53%

depending on Δt_h, Δt_o and λ. With low Δt_h and λ and high Δt_o values, the robots will be able to complete nearly all the tasks as expected. In contrast, with high Δt_h and λ and low Δt_o values, the robots are likely to fail completing many tasks as indeed it is the case here. It is observed that with $\Delta t_h = 30$ seconds, the robots are able to complete tasks effectively even if the value of λ is increased from 2 to 10 while this is not the case for $\Delta t_h = 300$ seconds. Thus, as Δt_h is increased, the robots can handle less tasks. When the parameter Δt_o is increased, the robots may have much time to handle the waiting tasks. Hence, the number of the timed-out tasks will decrease. Finally, normalized pairwise distance $\bar{M}_3$ varies between 7.1% and 11.2%. For example when $\bar{M}_1$ goes from 4.5% to 10.8%, $\bar{M}_3$ goes from 8.2% to 11.2% - meaning that the distance between the caller and assisting robots will increase from 10 meters to 60 meters. Thus, there is a trade-off between having a large number of assisting robots and low computational and communication requirements. For example, the selection of particular assisting robot can be ignored since even with $\bar{M}_1 = 10\%$, each robot selects out of 3-4 robots - a relatively small number. However, with near full-connectivity, the robots will have to select a particular robot out of 40-49 robots, such an assumption will no longer hold and percentage of time-outs will increase accordingly.

Table 4.3. The effects of increasing various parameters.

	Measure		
Parameter	$\bar{M}_1$	$\bar{M}_2$	$\bar{M}_3$
$\Delta t_h \uparrow$	–	$\uparrow$	$\uparrow$
$\Delta t_o \uparrow$	–	$\downarrow$	–
$c \uparrow$	$\downarrow$	$\uparrow$	$\downarrow$
$\lambda \uparrow$	–	$\uparrow$	$\uparrow$
Packedness $\uparrow$	$\uparrow$	$\downarrow$	$\downarrow$

As seen in Table 4.2, the performance is also dependent on robots' packedness. The more packed robots are, the higher will be the density of the assistance network as measured by $\bar{M}_1$. Since the density and the level of packedness both increase, the robots can handle more tasks with lower pairwise distances. The percentage of timed-

out tasks $\bar{M}_2$ varies between 0.8% to 50%, which is similar to the result in Table 4.1. The normalized pairwise distance $\bar{M}_2$ varies between 5.7% and 12.1%. The changes of Δt_h, Δt_o and λ values have the same effects on the task completion performance as mentioned above. When these results are all put together, the effects of increasing various parameters can be summarized in Table 4.3. As expected, network density increases with decreased cost parameter and increased robot packedness. Timed-out tasks decrease as all parameters decrease with the exception of Δt_o. Finally, normalized pairwise distance decreases as cost parameter and packedness both increase.

Table 4.4. Simulation results for varying proximity ρ_h in proximity-based assistance.

	Δt_h sec	30		300		30		300	
λ	Δt_o sec	60	180	60	180	60	180	60	180
	ρ_h meters	248				143			
2	$\bar{M}_1$%	10.4				4.7			
	$\bar{M}_2$%	13.8	12.3	27.3	20.4	38.1	35.9	49.5	43.5
	$\bar{M}_3$%	5.7	5.8	6.2	6.3	4.2	4.2	4.3	4.4
10	$\bar{M}_1$%	10.4				4.7			
	$\bar{M}_2$%	14.0	12.7	55.5	48.5	38.5	36.6	70.6	66.2
	$\bar{M}_3$%	6.0	6.1	7.4	8.0	4.3	4.3	4.7	4.9

4.6.2. Comparative Study: CPG-Based vs Proximity-Based Assistance

Next, the performance of the system with CPG-based assistance networks is compared with proximity-based assistance where the network coordinator updates the topology based purely on proximity considerations. Namely all robots within a range ρ_h are designated as potential assisting robots. Of course, the selection of ρ_h is critical to performance. If it is too small, then the robots may end up not having any robots that can assist them. On the other hand, if its value is large, this will possibly lead to a completely connected assistant network which has its own problems as discussed in Section 4.4. For fair comparison, its value is selected so that the compared networks have similar network density. For example, for $c = 2$, average density in CPG-based

Table 4.5. Simulation results for varying packedness in proximity-based assistance.

λ	Packedness	Low				High			
	Δt_h sec	30		300		30		300	
	Δt_o sec	60	180	60	180	60	180	60	180
	ρ_h meters	311				161			
2	$\bar{M}_1\%$	5.5				9.5			
	$\bar{M}_2\%$	9.5	8.3	24.7	17.5	22.6	21.0	34.1	27.9
	$\bar{M}_3$	8.1	8.2	8.7	8.9	3.5	3.6	3.8	4.0
10	$\bar{M}_1\%$	5.5				9.5			
	$\bar{M}_2\%$	9.9	8.7	55.9	49.4	23.0	21.7	58.7	51.9
	$\bar{M}_3\%$	8.5	8.6	9.9	10.4	3.7	3.8	4.8	5.3

assistance networks is 10.8. In order to have a similar level of density, $\rho_h = 248$. With this in mind, it is varied as $\rho_h \in \{248, 143\}$ meters for various cost scenarios and $\rho_h \in \{311, 161\}$ for various packedness scenarios.

The results are as seen in Tables 4.4-4.5 respectively. It is observed that in CPG-based assistance networks, the percentage of timed-out tasks is considerably lower as compared to proximity-based assistance. In turn, the average pairwise distance $\bar{M}_3$ is slightly lower – as proximity-based assistance is programmed only with this in mind whereas the topology of CPG-based assistance networks takes both the number of neighbors and the distance into consideration. Robots may be willing to assist even if they are distant from each other. Hence, pairwise distance in the CPG-based assistance turns out to be higher than that of the proximity-based. For $\Delta t_h = 30$ sec, the percentage of timed-out tasks in the CPG-base is about 1%, which does not depend on the value of λ. On the other hand, that of the proximity-based varies from 9% to 30%.

Figure 4.5. The experimental setup of a patrolling mission. It is assumed that as robots start moving around, they encounter dynamic and cooperative tasks which they need to handle.

4.7. Experiments

4.7.1. Experimental Platform

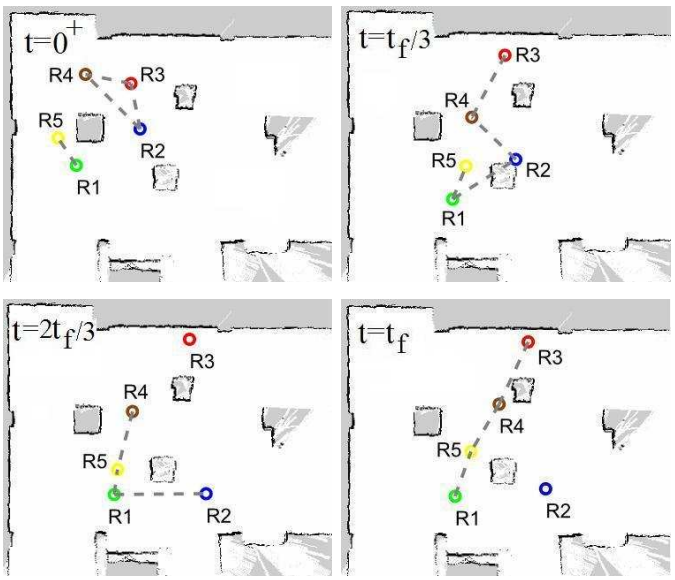
This section contains a report of experimental evaluation of the proposed approach with a multirobot system composed of five Turtlebot robots that are operating in a workspace of 6m $\times$ 6m as shown in Figure 4.5. All the robots are equipped with on-board processing, gyro, and Hokuyo laser range scanner with 4m field of depth and encoders. They move with maximum speed around 0.1 meters/second in the workspace. The robots have been programmed using Robot Operating System (ROS). The map of workspace is built using gmapping package in ROS with one of the robots prior to the experiment. This map is then shared by all the robots. Localization is achieved via the Adaptive Monte Carlo Localization package that is available in ROS. The control software has a special novel architecture that extends the sense-act loop of control architectures to sense-communicate-act paradigm. [103]. For the interested reader, the new control architecture is presented in summary in Appendix E. Two new modules are introduced: communication and network update. The communication module enables the robots to communicate with potential assisting robots as deemed by the current assistance network. The network update module has different function-

alities depending on the running robot. If a robot has the additional role of being the network coordinator, periodically it receives each robot's information, updates the network topology and informs all the robots accordingly. Otherwise, each robot simply relays its position information to the coordinator and waits for the updated network topology. For patrolling, a modified version of feedback-based navigation is used [104]. For this, MPFR library is used to operate on the big numbers while implementing the navigation controller [105]. All the robots also have a task detection module that is responsible for detecting encountered tasks. This module should be implemented depending on the application and the type of tasks that may be encountered. For example, in case of object moving tasks, the robot should be able to detect this task accordingly. Finally, they all have a task handling module as described in Section 4.2. the corresponding algorithm for this module is given in detail in Appendix B.

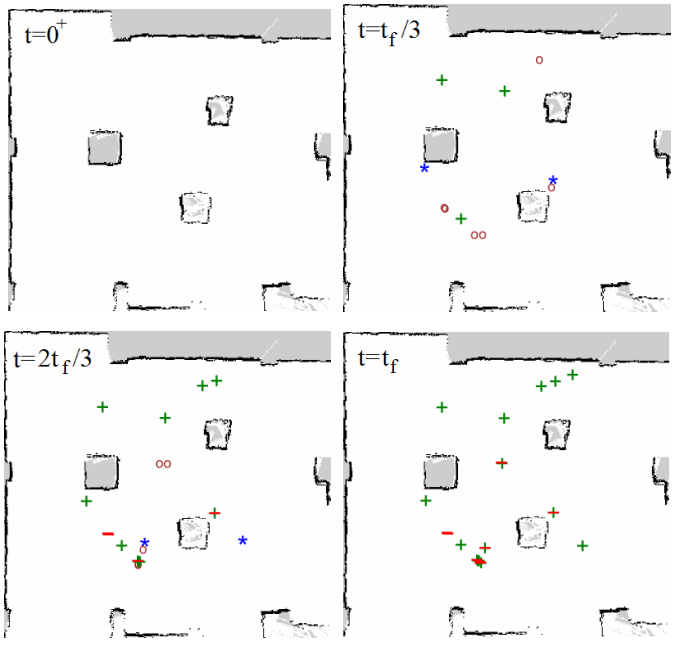
4.7.2. Experimental Results

Similar to the simulations, as they are moving around in this area, they dynamically encounter tasks that require cooperation. Robot 3 (with red cone) is assigned to be the network coordinator. It evolves the assistance network with the cost parameter $c = 1.5$. The network update period $T_g = 15$ seconds.

Again, as our focus is on overall system performance, as the robots are moving around, the tasks they encounter are generated synthetically by a Poisson process with parameter $\lambda = \{3, 6\}$. These values result on average 7 tasks ($\lambda = 3$) and 23 tasks ($\lambda = 6$) per minute during each mission. The task duration is $\Delta t_h = 10$ seconds and maximum allowable waiting time is $\Delta t_o = 20$ seconds. The robots participate in 5 mission-task scenarios for each λ. Four sampled instances from one of these mission-task scenarios is as shown in Figure 4.6. As robots start out at time 0^+, the assistance network is formed. The robots have not encountered any tasks yet. About one third of their mission, the assistance network has evolved accordingly with 3 tasks completed. The robots are engaged in 2 tasks while 6 new tasks are waiting to be handled. It should be noted that as the workspace has an area of 36 m^2, nearby tasks are seen as one in the figures. At time $2t_f/3$, two of previously waiting tasks are now completed.



(a)



(b)

Figure 4.6. A sample experiment - a 5-robot mission with $\lambda = 6$. (a) Assistance network evolution. The circles indicate robots while the assistance network g is shown by the dotted lines; (b) Encountered tasks and their statuses. Each completed task is shown by a green plus while each timed out task is shown by red minus. Each task being handled is indicated by blue star while red small circles indicate waiting tasks.

Altogether 10 tasks have been completed. 5 new tasks are waiting while 2 tasks are being handled. Finally at the end of the mission, 15 out of 23 encountered tasks have been completed.

Table 4.6. Average percentage of timed-out tasks.

	Robots				
λ	R1	R2	R3	R4	R5
3	0	0	0	0	0
6	18.5	35.8	16.7	49	47.5

The individual timed-out task percentages for $\lambda = 6$ are as presented in Table 4.6. These vary between 17% to 49% - which imply about 15% variation in performance with respect to average timed-out percentage of 29.9%. Statistical performance measures are shown in Table 4.7. The network density turns out to be 43%. Although this value seems to be high compared to the simulations with 50 robots, it corresponds to average of 1.7 potential assisting robots in the 5-robot experiment. This is in contrast to average of 2.3 robots for a density of 4.8% of density in 50-robot simulations. With $\lambda = 3$, all the tasks are completed without any time-outs. Interestingly, when the number of tasks increases with increased $\lambda = 6$, about a third of the tasks cannot be completed. This suggests that the cost parameter can be used to control the density of the assistance network.

Table 4.7. Experimental statistics.

λ	3	6
$\bar{M}_1\%$	43.4	43.2
$\bar{M}_2\%$	0	29.9

4.8. Conclusion

This chapter considers the problem of determining the assisting robots in tasks requiring the cooperation of robot pairs. It presents a novel approach based on assistance networks that designates potential helpmates in an implicit manner based on its topology. Each robot – when in need - seeks to find an assisting robot among its immediate neighbors in the assistance network in a decentralized manner. As the robots are moving around, the assistance network topology needs to be continually updated. This update is done by a coordinator in a centralized manner - following a methodology similar to that of centralized network topologies. Extensive simulation results indicate that tasks can be handled effectively depending on occurrence of tasks and their associated constraints. As expected, as the number of encountered tasks increases, the robots may fail to complete all the tasks. While this may be offset by increasing the assistance network density via decreasing the cost parameter in the payoff functions, there is a trade-off between increased number of potential assisting robots and computational and communication requirements as measured by the pairwise distance between each robot and its assisting robots. Indeed, with decreased cost parameter values, while time-out percentages decrease, pairwise distance between robots' and their potential assisting robots increase. The practical applicability of the proposed approach is tested in a series of experiments with a team of five robots.

5. COALITION FORMATION GAMES FOR DYNAMIC MULTIROBOT TASKS

Multi-robot systems operating in unknown environments need to handle tasks they encounter dynamically [68, 73]. Some of these tasks may require a multitude of different resources in order to be successfully completed. If the robots vary in their sensory, actuation and computational capabilities, they may not individually possess all of the required resources to accomplish a task. In case of insufficient resources, a new subgroup of robots needs to be formed - considering the whole robot team and having the required resources. While the type of tasks and the resources they require will vary depending on the application, in all, the question of assembling the teams of robots – commonly referred to as coalitions - capable of doing the tasks to be addressed effectively. This is a challenging task. As tasks are dynamic, they occur at unpredictable places or times which implies that the robot teams cannot be assigned a priori. If the coalitions are formed with resources far surpassing the requirements, then it may not be possible to accomplish other encountered tasks. Furthermore, robots' locations will need to be considered since choosing far away robots may lead to unacceptable task start periods.

In this chapter, we consider the problem of coalition formation for dynamic tasks with particular resource requirements. In a taxonomy of MRTA problems based on three orthogonal dimensions - namely single-task (ST) vs multi-task robots (MT) depending on whether each robot is capable of executing single or multiple tasks at the same time; single-robot tasks (SR) vs. multi-robot tasks (MR) depending on whether a task can be completed by a single robot or several robots (a coalition) and instantaneous assignment (IA) vs. time-extended assignment (TE) depending of whether only current tasks or future tasks are considered [73], this problem is an instance of the ST-MR-IA problems. As it is known to be strongly NP-hard, approximate solutions with an emphasis on computational feasibility and practical applicability need to be developed [68].

This chapter proposes a coalition formation games (CFG) based approach to forming coalitions by considering both the task requirements and the resources and capabilities of the robots together. We assume that each robot can participate in one task only and thus can be a member of only one coalition. We model the coalition formation problem as a coalitional game and propose a centralized coalition formation (CCF) algorithm. Using CCF, a task coordinator autonomously forms coalitions for each pending task a coalition while maximizing the coalitions' values in terms of resource satisfaction while accounting for moving to task site. It is shown that independent disjoint coalitions are capable of performing these tasks.

The outline of this chapter is as follows: First, related work is reviewed in Section 5.1. Next, coalitions and tasks are formulated in Section 5.2. The task coordinator is explained in Section 5.4 followed by a discussion of coalitional stability in Section 5.5. Extensive simulations provide insight into task performance in a multi-robot system consisting of 50 robots in Section 5.6.

5.1. Related Literature

We review coalitions in multi-agent systems, game theory and robotics.

5.1.1. Multi-Agent Systems

Coalition formation is a key topic in multiagent systems where the goal is to find a coalition structure that maximizes the sum of the values of the coalitions. It is an instance of a set partition problem which is known to be a NP-complete [106]. As the exorbitant number of coalition structures does not allow exhaustive search for the optimal one, the focus has been on finding a coalition structure via a partial search with guaranteed proximity to the optimum. It has been shown that the number of nodes that need to be searched for establishing a bound is required to be greater than a calculated threshold [79] where an algorithm that establishes a tight bound within this minimal amount of search is also presented. In [107], a greedy heuristic is used to yield a coalition structure that is provably within a bound the of best - limiting the

coalition sizes. This approach is general as it can be applied in environments that are not necessarily super-additive and agents are assigned tasks in a distributed manner .

5.1.2. Game Theory

As the task allocation problem among the agents is mapped into the problem of the formation of coalitions, coalition formation games (CFG) provides a suitable analytical tool for studying the formation of coalitions among a number of players [108, 109]. CFG is branch of game theory investigating algorithms to study how coalitions form in cases where superaddivity - namely where any two disjoint coalitions, when acting together, can get at least as much as they can when acting separately - does not hold. Hence, grand coalition is no longer optimal [110]. Thus, in coalition formation, the focus is on split and merge rules that transform partitions of players with a concept of stable partitions [111]. Coalition stability is related to the type of membership changes allowed. In hedonistic games, only one individual agent is considered to change its coalition at a time [112]. With this, the notions of stability vary from contractual individual stability to individual stability to Nash stability. In more general settings, groups of agents are allowed to change their coalitions simultaneously. In this case, the definition of stability is based on the set of allowable coalitional changes. In all, a comparison operator that orders the sets of coalitions is defined. This comparison operator is either based on the the coalition value that quantifies the worth of a coalition or the individual players' payoff [113].

5.1.3. Robotics

In robotics applications, the problem is generally viewed as an instance of optimal assignment of a set of tasks to the robots with respect to an overall performance objective while taking their individual constraints into consideration. The objectives encode demands, resources and gains if possible [81]. The decision-making varies from being centralized to being distributed [76, 77]. Centralized approaches are simpler to implement, but become practically infeasible as the number of tasks and robots increase. On the other hand, while distributed approaches are advantageous with

respect to reliability and scalability, the efficiency of the resulting solutions are harder to ensure. In practice, systems may not conform to a strict centralized/decentralized dichotomy and may contain both elements [78].

Most applications assume that all the tasks are known initially. In these problems, the tasks are allocated iteratively. In this perspective, many heuristic algorithms have been developed [73]. For example, in [114], multi-agent task allocation algorithm [107] is modified in order to accommodate the fact that resources of a coalition are not collectively available and cannot be redistributed. An anytime algorithm is shown to have bounded solution with a minimal search [83]. Improved solutions are obtained such a polynomial time dynamic programming approach in the case of agents of different types being indistinguishable or solutions to be within a constant factor of the optimal utility the population of coalitions are restricted [84]. Two natural greedy heuristics are extended via a new greedy heuristic that considers the expected loss of utility due to the assigned robots and task as an offset and uses the offset utility for task assignment [74].

Distributed approaches solve the constrained optimization problem in a decentralized manner [86]. For example, in a class of problems known as distributed constraint optimization problems (DCOPs), each robot or group controls one set of variables and together they have the joint goal of maximizing a global objective function [44,87]. For example, distributed efficient greedy algorithms require robots have only local knowledge about tasks and resources. In best response algorithms, each robot or group reacts on the basis of local knowledge so that there is no need for tree-like communication [88]. In social networks, the robots are distributed over a given network and only neighboring agents are allowed to work together in a task [89, 90]. However, in these works, the network topology is assumed to be given a priori and fixed. As such, the robots face with the difficult problem of searching a factorial sized space for an optimal strategy which naturally has adverse effects on performance [91].

Alternatively, in multi-objective approaches, a set of objective functions is simultaneously optimized. In general, multiobjective optimization may lead to different

solutions as compared to optimizing with respect to a single objective [44]. One of the most popular approaches that falls in this category is the market based strategy [95]. Market based task allocation combines the efficiency of a centralized approach (the auctioneer decides with overview of the situation) with advantages of distributed approaches (much of the calculation is done by the individual robots preparing their bids) [92]. An auctioneer announces tasks, and robots make bids, indicating their cost or utility to deal with the tasks and assigns tasks. The auctions can be conducted either in centralized or distributed manner. The centralized auctioneer provides a framework for considering optimization of a global objective and can be combinatorial [96] or greedy [97]. In case of constrained resources, a greedy optimal solution is proposed via a leader follower coalition method where coalition utility is maximized for every assigned task [85]. The auction process may be split into task and robot auctions as is done in the Double Round Auction approach [115]. With increased number of robots, market based systems are not always the most appropriate approach. This is because they use the auctioneer as a central decision maker, use an explicit assignment of individuals to jobs, and need some form of global or at least long distance communication. All of these elements reduce the scalability and robustness of the system, and conflict with the distributed and purely local way of working of many systems [67,116]. Distributed auction approaches are derived from centralized auctions using regional opportunistic centralization [93, 98–100]. However, the locality of decisions may block idle, but remote robots coming to assistance.

A variety of different approaches have also been proposed. An approach based on token passing avoids the large communication bandwidth requirement from which market-based approaches suffer [101]. Two distributed and efficient approaches that rely on simple interaction through light signals or more advanced gossip-based communication scheme to announce task requirements among the robots [116]. However, these are applicable only in very simple task scenarios. The number of robots is tuned in order to improve the group efficiency using a biology inspired method [117] or energy optimization [118].

Finally, let it be noted tasks may be dynamically encountered. To our knowledge,

ST-MR-IA problems with dynamic tasks have been relatively less considered. The iterated type algorithms need to modified in order to accommodate dynamic multi-robot tasks.

5.2. Coalitions & Tasks

A multirobot system consists of a set of $\mathcal{R} = \{1, \ldots, r\}$ robots. We assume that each robot $i \in \mathcal{R}$ has radius $\rho_i \in \mathbb{R}$ and is uniquely identifiable. It is associated with a time-varying state $b_i \in \mathbb{R}^2$. The robots are assumed to be heterogeneous which imply that they vary in their resources. Assuming there are N_r different types of resources, each robot i is also associated with a resource vector $r_i = [r_i(1), \ldots, r_i(N_r)]^T$ with $r_i(j) \geq 0$, $j = 1, \ldots, N_r$ where $r_i(j) \in R^{\geq 0}$ denotes the amount of j-th resource that robot i has. If robot i does not have any of resource j, then $r_i(j) = 0$.

5.2.1. Coalitions & Resources

A coalition C_c is a non-empty subset of $\mathcal{R}$. The set $\mathcal{R}$ is known as the grand coalition while a coalition with just one robot is referred to as singleton coalition. A collection in $\mathcal{R}$ is any family of $C = \{C_1, \ldots, C_s\}$ of mutually disjoint coalitions. If $\cup_{c=1}^{s} C_c = \mathcal{R}$ where s is the number of coalitions, then C is called a partition [111]. If partitions evolve over time, then they become time varying as $C(t) = \{C_1(t), \ldots, C_{s(t)}(t)\}$ with the number of coalitions $s(t)$ becoming time-varying as well. Hence coalitions are not static, but rather changing over time.

Each coalition C_c is associated with a set of resources with possible types of the resources known. The resource vector is denoted $R_c = [R_c(1), \ldots, R_c(N_r)]^T$ with $R_c(j) \geq 0$, $j = 1, \ldots, N_r$. It is assumed that resources are additive - namely

$$R_c(j) = \sum_{k \in C_c} r_k(j)$$

Furthermore, each coalition has a leader (head). The leader coordinates the coalition. The leader may change over time since the coalitions may evolve over time. The rules for selecting leader are as follows:

- The leader does not change unless it leaves the coalition.
- If the leader leaves the coalition or the coalition does not have a leader, the robot with the smallest robot ID value becomes the coalition leader.

5.2.2. Tasks & Resources

It is assumed that the workspace is bounded by radius ρ_0. As a robot or a coalition of robots is moving around this workspace, it will come across a number of tasks. There is no a priori information regarding their spatial locations or when they are likely to encounter one. Each task T is defined by the following:

- t_1 - Robot ID of the robot that has encountered the task.
- t_2 - Time of encounter.
- t_3 - Location of the task
- t_4 - Vector of the required resources defined as $t_4 = [\tau_1, ..., \tau_{N_r}]$. If resource l is not required for the task t, then $\tau_l = 0$.
- t_5 - Maximum allowable waiting period for handling the task. Failure to do so results in task time-out.
- t_6 - Coalition leader ID in the coalition that is handling this task.
- t_7 - Time when the task starts being handled

- t_8 - Status of the task defined as:

$$t_8 = \begin{cases} 0^- & \text{if task is waiting in the coalition} \\ 0^+ & \text{if task is waiting in the coordinator} \\ 1^+ & \text{if task is being handled} \\ 1^- & \text{if a coalition is assigned, but task has not started yet} \\ 1 & \text{if task was completed} \\ -1 & \text{if task could not be completed} \end{cases}$$

When a task is initiated, $t_8 = 0^-$ - which indicates the task has been just encountered. When resources are found to be insufficient, $t_8 = 0^+$. The case $t_8 = 1^-$ indicates that a coalition has been assigned, but all the coalition members have not reached to the task site.

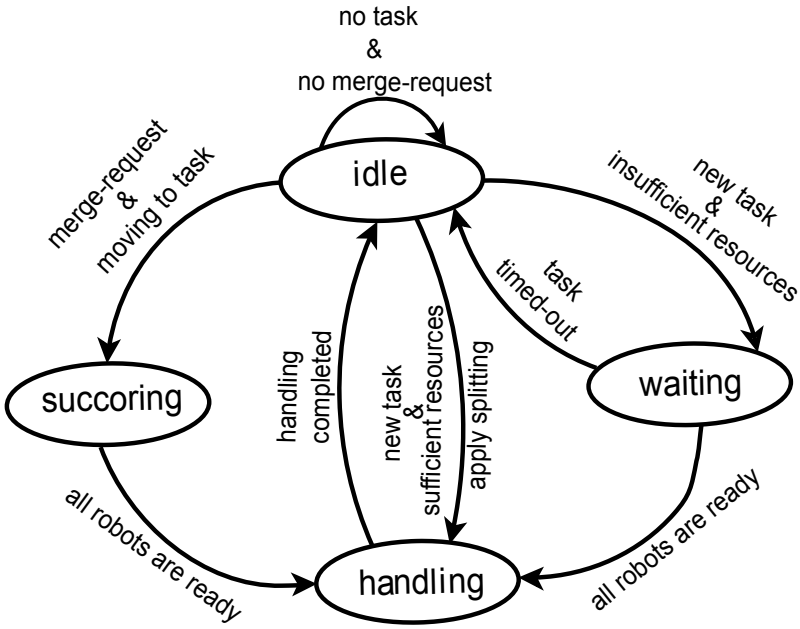


Figure 5.1. Task automaton of a coalition.

5.3. Task Automaton

Every time a coalition encounters a task, the leader activates the task automaton. The task automaton is associated with four states: 'idle', 'handling', 'succoring' and

'waiting' as shown in Figure 5.1. Normally, the coalition is in idle state. When a coalition encounters a task, it creates an instance of task T. It then checks for the following:

Rule 1: If the coalition is idle and has enough resources for this task - namely $\forall \tau_j, j = 1, \ldots, N_r \quad R_c(j) \geq \tau_j$ it goes into the handling state.

In the handling stage, before starting with the task, the coalition checks for excessive resources and attempts to be as lean as possible - namely

Rule 2: If the coalition has surplus of resources, $\forall \tau_j, j = 1, \ldots, N_r \quad R_c(j) - \tau_j > 0$ then split as much as possible.

In case resources are excessive, then without contacting the coordinator, the leader splits its coalition so that some robots may be free to participate in other coalitions. If the leader is taken out, the coalition then chooses the next leader. In all cases, it then starts handling the task.

If the coalition does not have enough resources, then it reports this task to the task coordinator and switches to waiting state - namely:

Rule 3: If the coalition is idle, but does not have enough resources, then it then reports the task to the task coordinator and waits for a response.

The coalition remains at the location where it encounters the task - waiting for a response from the task coordinator. If the task coordinator does not convey an assignment within a certain time, the coalition goes into idle state again.

If the task coordinator conveys a particular task, member robots move to the task site. During this movement, the coalition will be in succoring state. Once all the robots reach the task location, the coalition goes into handling state. At the end of task completion, the task information is transmitted to the task coordinator.

Finally, the coalition may encounter new tasks while it is not in the idle state. In this case,

Rule 4: If the coalition is not idle, the coalition leader informs the coordinator of this task.

5.4. Task Coordinator

The task coordinator is responsible for forming the robot coalitions capable of performing each reported task [6] . Let $\mathcal{T} = [T^1, \ldots, T^{n_t}]$ be the set of reported tasks which are waiting (either in the coalition or in the coordinator) where n_t is their total number. As tasks get completed or timed-out, they are removed from this list. As tasks are dynamic, they occur at unpredictable places or times which implies that n_t is changing over time.

The task coordinator is associated with a time-varying state $C(t)$ that designates the current set of coalitions $C(t) = \left\{C_1(t), \ldots, C_{s(t)}(t)\right\}$ where $\cup_{c=1}^{s(t)} C_c \subseteq \mathcal{R}$. The set $C(t)$ is updated periodically and remains the same between updates. In each update, the task coordinator finds a new map $\alpha_t : \mathcal{T} \rightarrow C(t) \cup \emptyset$. If $\alpha(T) = \emptyset$, this means that a coalition with sufficient resources cannot be formed. The map α_t is defined based on coalition values via coalition formation games.

5.4.1. Coalition Value

The coalition value describes the overall utility each coalition $C_c \in C(t)$ receives associated with task T. As such it is comprised of three terms. The first term measures whether the coalition has sufficient resources to complete the task as measured by $\sum_{j=1}^{N_r} \gamma(\tau_j - R_c(j))$ where

$$\gamma(x) \quad = \quad \begin{cases} 0 & x <= 0 \\ x^2 & x > 0 \end{cases} \tag{5.1}$$

[6] This may be one of the robots with the additional task of being a coordinator.

Secondly, robots that utilize that resources to greater extent are preferred. This is measured by the term $\sum_{j=1}^{N_r}\left(1-\frac{R_c(j)}{\tau_j}\right)^2$. Finally, the proximities of coalition members to the task site are also taken into consideration - as nearby free robots will be preferred. This is measured by $\sum_{l\in C_c}\frac{\delta_{i,T}}{2\rho_o}$

All these considerations are taken into account:

$$v(C_c,T) \;=\; \frac{1}{1+\beta(C_c,T)} \tag{5.2}$$

where

$$\beta(C_c,T) \;=\; w_1\sum_{j=1}^{N_r}\gamma(\tau_j-R_c(j))+w_2\sum_{j=1}^{N_r}\left(1-\frac{R_c(j)}{\tau_j}\right)^2+w_3\sum_{l\in C_c}\frac{\delta_i}{2\rho_o}$$

with the value δ_i is the distance between robot i and the location of task T. The parameters w_1, w_2 and w_3 are relative weighting parameters of resource satisfaction, resource excessiveness and site proximity.

5.4.2. Comparing Partitions

In this case, the comparison of different partitions is based on comparing coalitions using their values.

Definition 5.1. *A comparison relation $\triangleright$ is defined for comparing two collections A and B that are partitions of the same set $\mathcal{R}$. If $A \triangleright B$, then the partition A is preferred to partition B.*

Note that each comparison relation is used only to compare partitions of the same set of players. So partitions of different sets of players are incomparable.

Various criteria can be used as comparison relation between partitions [111]. An adequate individual value order that can be used is the Pareto order. Let the $\varphi_i(C_c)$

denote the payoff of each robot for being in coalition $C_c \in C$. Let this be equal to

$$\varphi_i(C, T) = \varphi_i(C_c, T) = v(C_c, T)$$

The Pareto-order is defined as:

$$A \triangleright B \Leftrightarrow \varphi_i(A, T) \geq \varphi_i(B, T) \, \forall i \in \mathcal{R}$$

with at least one strict inequality ($>$) for one robot k.

5.4.3. Coalition Formation Games

This process is modeled based on coalition formation games using the Pareto order as a comparison relation. However, even with a modest robot population size, the number of partitions - namely the Bell number - is exorbitant. For example, for $r = 10$, the number of partitions is 115975. As such, the number of partitions is too large to allow for exhaustive search for the optimal one [79].

In this framework, we resort to coalition formation games. We model the coalition allocation problem as a coalitional game and propose a centralized algorithm for coalition formation. Using the Pareto order as a comparison relation, we propose a centralized coalition formation algorithm based on two rules called 'merge' and 'split' that allow to modify a partition C of $\mathcal{R}$ as follows [119]:

- **Merge:** $\{C_1, \ldots, C_k\} \rightarrow \cup_{j=1}^{k} C_j$ where $\cup_{j=1}^{k} C_j \triangleright \{C_1, \ldots, C_k\}$.
- **Split:** $\cup_{j=1}^{k} C_j \rightarrow \{C_1, \ldots, C_k\}$ where $\{C_1, \ldots, C_k\} \triangleright \cup_{j=1}^{k} C_j$.

The coordinator uses merge and split operations on the existing coalitions. As such, different coalitions are allowed to interact - taking the decision to merge or split based on the comparison relation $\triangleright$. With Pareto order, the task coordinator decides to merge or split coalitions only if at least one coalition is able to strictly improve its

individual value through this process without decreasing the other coalitions' values. Therefore, the merge rule by Pareto order is a binding agreement among the robots to operate together if it is beneficial for the tasks.

5.4.4. Coordinator Algorithm

The coordinator periodically considers all the tasks waiting for coalitions and starts a coalition formation game as shown in Figure 5.2. In this algorithm, only idle or waiting coalitions are considered. Merge-split rules are iteratively applied until no all the coalitions associated with all the tasks stabilitize. Note that as a result, some tasks may be associated with empty set – which implies that the coordinator cannot find a coalition capable of performing that particular task.

5.5. Partition Stability & Convergence

Stability is related to the type of coalition membership changes allowed [119]. If group movements are allowed, then robots may change coalitions in arbitrary groups. Allowable membership changes are defined by a defection function $\mathcal{D}$ that associates with each partition C of $\mathcal{R}$ a group of collections in $\mathcal{R}$. Namely it consists of all collections $C = \{C_1, \ldots, C_l\}$whose players can leave the partition S by forming new and separate group of players $\cup_{j=1}^{l} C_l$ divided according to the collection C. A partition $C = C_1, ..., C_l$ of $\mathcal{R}$ is $\mathcal{D}$-stable if no group of robots is interested in leaving C when the robots who leave can only form the collections allowed by $\mathcal{D}$. Mathematically,

Definition 5.2. *$\mathcal{D}$-Stability Assume a comparison relation $\triangleright$ and a defection function $\mathcal{D}$. A partition P is called $\mathcal{D}$-stable if $\forall\ C \in \mathcal{D}(P)$ such that $C[P] \neq C$, $C[P] \triangleright C$.*

The most general case is with defection function $\mathcal{D}_c$, where for each partition P, $\mathcal{D}_c(P)$ is the family of all collections in $\mathcal{R}$. As such, any group of robots can leave P and create an arbitrary collection in $\mathcal{R}$. However, in this case, it is known that value functions need to be superadditive – which is not the case here. An alternative definition is using a defection function $\mathcal{D}_{hp}$ - where for each partition P, $\mathcal{D}_{hp}(P)$ is the

1: **while** Change in partition C, repeat **do**
2: **while** Change in partition C, merge **do**
3: **for all** $T^k \in \mathcal{T} \mid t_8^k = 0^-$ or $t_8^k = 0^+$ **do**
4: **if** $t_8^k = 0^+$ **then**
5: Assign C_c with the highest $v(C_c, T^k)$
6: **end if**
7: Find a coalition C_d such that $C_c \cup C_d \triangleright C_c$
8: **if** $C_d \notin \emptyset$ **then**
9: $C_c = C_c \cup C_d$
10: **end if**
11: **end for**
12: **end while**
13: **while** Change in partition C, split **do**
14: **for all** $T^k \in \mathcal{T} \mid t_8^k = 0^-$ or $t_8^k = 0^+$ **do**
15: Find $i \in C_c$ such that $C_c - \{i\} \triangleright C_c$
16: $C_c = C_c - \{i\}$, $C = C \cup \{i\}$
17: **end for**
18: **end while**
19: **end while**

Figure 5.2. The algorithm representing the coordinator automaton.

Table 5.1. Simulation settings.

Parameter	**Value**
Mission duration	60 minutes
Number of robot types	10
Number of resource types	5
Number of task types	10
Time-out duration	$\{1, 2\}$ minutes
Task rate λ for each robot	$\{5, 10, 20\}$ tasks per hour
Range of resource values of tasks	$[30, 50]$ units
Range of resource values of robots	$[5, 10]$ units
Range of handling duration	$[2, 5]$ minutes

family of all P-homogeneous partitions in $\mathcal{R}$. In this case, the defection function $\mathcal{D}_{hp}$ allows the players to leave the partition P only by means of merges or splittings - albeit with multiple applications. A partition is $\mathcal{D}_{hp}$-stable if and only if it is the outcome of iterating the merge and split rules. This is an immediate consequence of Theorem 5.3 as presented in [119] that guarantees the existence of a $\mathcal{D}_{hp}$-stable partition.

Theorem 5.3. *[119] A partition $P := P_1, ..., P_k$ of $\mathcal{R}$ is $\mathcal{D}_{hp}$-stable if and only if the following two conditions are satisfied:*

(i) $\forall i \in \{1, ..., k\}$ *and for each partition* $C_1, ..., C_l$ *of coalition* $P_i \quad v(P_i) \geq \sum_{i=1}^{l} v(C_i)$

(ii) $\forall L \subseteq \{1, ..., k\}$ $\sum_{i \in L} v(P_i) \geq v(\bigcup_{i \in L} P_i)$

Since each rule application increases the values of the coalitions with an assigned task, the process terminates. This is stated in Theorem 5.4. Hence the resulting partition will be $\mathcal{D}_{hp}$-stable.

Theorem 5.4. *[111] Suppose that $\triangleright$ is a comparison relation. Every iteration of the merge and split rules terminates.*

5.6. Simulations

Extensive simulations have been conducted with 50 robots placed in a workspace of radius 100 meters. The robots are cylinder shaped with radii 15 cm and can move with maximum speed of 0.6 meters/second. General simulation settings are as presented in Table 5.1. The robots or coalitions - if formed - are assumed to be in a patrolling mission in this workspace while in idle state. Each robot is one of 10 types - categorized according to its resources.

As tasks are encountered, each coalition invokes its task automaton. Whereas, the coordinator applies its automaton every 10 seconds. Hence, the coordinator considers the waiting tasks every 10 seconds. Each task may become one of 10 types - again categorized according to required resources as given in Table 5.1. Comparing robot's resources and tasks' requirements, the populations of coalitions capable of accomplishing the encountered tasks are expected to vary between 3 to 10 robots. Time-out periods are taken to be 1 or 2 minutes - which imply relatively fast response times are required. Tasks that fail to start in this period cannot be completed. Once a task is completed, the respective coalition starts moving in a direction as determined by the coalition leader.

As our focus is on the formation of coalitions with sufficient resources, it is assumed that tasks are generated as a Poisson process with parameter $\lambda \in \{5, 10, 20\}$ tasks per hour. As the average number of tasks will increase with increased λ value, hence making the overall missions more difficult and challenging. Maximum allowable waiting period for handling the task is varied as $t_5 = \Delta t_o \in \{60, 120\}$ seconds.

At the beginning of each simulation, 10 different robot types and 10 different tasks are generated - with capabilities and task resource requirements falling randomly in the range as shown in Table 5.1. Sample robot and task types are as shown in Table 5.2 and Table 5.3. The handling duration of a detected task is also randomly generated from the range of handling duration. The parameters in the coalition value function are selected as $w_1 = 1$, $w_2 = 1$, and $w_3 = 1$.

Table 5.2. Robots' resources for a sample mission.

	Resources				
Robot Type	$r(1)$	$r(2)$	$r(3)$	$r(4)$	$r(5)$
1	6	5	5	9	8
2	7	8	8	8	7
3	7	5	6	7	8
4	7	8	5	5	7
5	5	6	8	5	5
6	6	9	7	6	6
7	6	6	9	7	5
8	7	9	7	8	6
9	9	8	9	8	6
10	6	6	5	8	5

For each level of each level of task rates and time-out, 50 different missions are conducted with robots all start at random initial locations. Hence, 300 missions are generated. Sample spatial distributions of tasks encountered in a mission for different λ values are as shown in Figure 5.3 where there are 48 and 98 tasks respectively. Tasks that are successfully completed are shown in green while those that were timed-out are in red. When the number of tasks are relatively fewer ($\lambda = 5$), 94% of the 48 tasks are completed. As the number of tasks is increased ($\lambda = 20$), the percentage of the completed tasks goes down to 78% with 77 out of the 98 tasks being completed.

Statistical results are as shown in Table 5.4. The percentage of timed-out tasks varies between 4.1% to 25.1% depending on Δt_o and λ. With high Δt_o and low λ values, the coalitions are able to complete many tasks ther encounter. In contrast, with low Δt_o and high λ values, the tasks are likely to time out. Hence, with the increase in time-out durations, the number of handled tasks increases. Average number of coalitions decreases with the increase in λ. This is because with the low λ values, there are less encountered tasks, hence there is no much need for merging coalitions. The last row

Table 5.3. The required resources for the tasks for a sample mission.

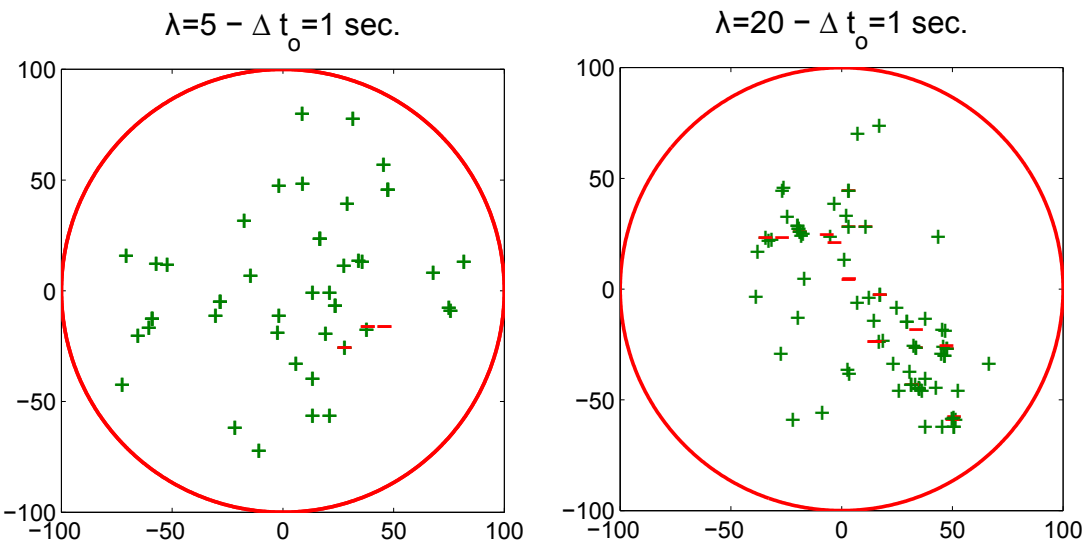
	Required resources				
Task Type	τ_1	τ_2	τ_3	τ_4	τ_5
1	47	44	42	42	47
2	44	38	32	41	36
3	44	48	42	33	48
4	41	40	47	49	46
5	39	33	35	45	44
6	34	42	46	38	30
7	35	40	48	41	41
8	34	33	33	37	31
9	48	44	30	32	41
10	35	35	41	31	31

in Table 5.4 gives the average number of decision-making of the coordinator. Since the coodinator considers the waiting tasks every 10 seconds, the maximum number of decision-making of the coordinator will be 360. According to the results, with the decrease in λ, the coalitions are able to handle their tasks without informing the coordinator.

Table 5.4. Simulation results.

	λ	5		10		20	
	Δt_o (min.)	1	2	1	2	1	2
Measures	Avrg. % of timed-out tasks	6.4	4.1	11.9	8.3	25.1	16.1
	Avrg. num. of tasks	44.1	43.5	61.2	60.3	90.1	82.1
	Avrg. num. of coalitions	15.5	12.3	10.7	11.7	8.3	9.4
	Avrg. num. of coor. dec.	44.7	52.2	79.1	102.2	149	179.8

The average number of iterations for the coordinator automaton in Figure 5.2



(a) (b)

Figure 5.3. Sample missions. The missions gets more challenging as the number of tasks increases. Robots encounter 48 tasks with $\lambda = 5$ while this number goes up to 98 tasks with $\lambda = 20$. Note that these tasks are not known a priori and can be completed only by a group of robots. With $\Delta t_o = 1$ minute, the completed tasks are as shown by green plus signs while timed out tasks are shown by red minus signs.

is presented in Table 5.5. The automaton has one main loop and two inner loops (merge and split). Taking the size of all possible partitions into account, the number of iterations for these loops is considerably low, thereby finding a partition for a given set of waiting tasks quickly.

5.7. Conclusion

This chapter considers the cooperative tasks requiring a set of resources. As the effective formation of robot teams endowed with these resources is crucial, the focus is on effective coalition formation. A task coordinator - similar to network coordinator – determines coalitions and assigning tasks. Its process is modeled as a coalition formation game where groups of robots are evaluated together in regards to each task's required resources and cost of forming a coalition. As new tasks are encountered, coalitions merge and split, hence the resulting coalitions are capable of doing these

Table 5.5. Average number of iterations (main loop - merge loop - split loop).

		Δt_o seconds	
		60	120
λ	5	1.47-2.7-1.7	1.44-2.57-1.68
	10	1.3-1.98-1.49	1.26-1.84-1.43
	20	1.2-1.59-1.38	1.17-1.5-1.34

tasks. Since the number of iterations for finding a suitable partition is considerably low, the proposed approach can be applied on the real-time robotic applications.

6. CONCLUSION

This thesis is concerned with multi-robot systems where robot cooperation is crucial to performance. The cooperation problem is examined from three critical and related aspects.

First, the problem of achieving effective communication topologies is considered. Two alternative approaches - namely decentralized or centralized network topologies - are proposed. These approaches define how network may form and evolve over time based on network related payoff functions and pairwise games. While all-to-all network, as is assumed in most works, is the simplest approach, it faces serious problems with the increase in the number of robots. Another most common approach is based on proximity in which only robots within a certain range are able to communicate with each other. However, distance may not always be best criterion in determining the network topology. In this manner, the contribution of both decentralized and centralized approaches is that the robots are able to manage the network topology while considering their network and/or task related objectives. In the decentralized approach, the robots - while engaged in their tasks - update the network topology periodically where the game interval is considerably smaller than the update period. The equilibrium networks are defined based on pairwise stability and pairwise Nash equilibrium. The pairwise game setting provides a general and practical scheme that can be employed by the robots simultaneously in order to contact other robots and update their local network topology depending on the exchanged information. As an application, we consider mutual link-based payoff functions and show that if each game is played with configuration states fixed to values corresponding to the game onset, the resulting networks are pairwise stable and thus acceptable to all the robots. Next, the case of centralized network topologies is considered. In this case, a network coordinator is held responsible for finding a network topology acceptable to all the robots. The robots communicate with a network coordinator periodically to get the updated network topology. In turn, the coordinator considers the individual payoff functions of all the robots, their current states and the current network simultaneously and finds

a network topology acceptable to all the robots. This is accomplished via centralized pairwise games which are basically iterated processes consisting of a sequence of game moves. With payoff functions satisfying local spillovers, convexity and strategic substitutes properties, each pairwise game is ensured of convergence to a pairwise stable network.

Next, the problem of determining the assisting robots in tasks requiring the cooperation of robot pairs is considered. Most previous works aim to find an explicit assignment of assisting robots. As this has been proven to be strongly NP-hard, approximate solutions with an emphasis on computational feasibility and practical applicability need to be developed. For this problem, a novel concept of assistance networks is introduced. In contrast to finding an explicit assignment of assisting robots, the assistance network designates potential helpmates in an implicit manner based on its topology. Each robot seeks to find an assisting robot among its immediate neighbors in the assistance network in a decentralized manner. As the robots are moving around, the assistance network topology needs to be continually updated. This update is done by a coordinator in a centralized manner. The coordinator process follows a methodology similar to that of centralized network topologies where acceptability is defined based on pairwise stability and pairwise Nash stability. Extensive simulation and experimental results indicate that tasks can be handled effectively depending on occurrence of tasks and their associated constraints. It is observed that as the number of encountered tasks increases, the percentage of timed-out tasks increases with a fixed cost parameter. This suggests an adaptive scheme for the cost parameter depending on missed tasks as well as allowable helpmate proximity. As part of future work, this issue can be investigated further. Also, assistance networks can be generalized to more complicated dynamic and cooperative task scenarios.

Finally, the more general case of tasks requiring a multitude of resources is considered. A task coordinator is held responsible for finding robot coalitions endowed with these resources with minimal cost. This process is modeled as a series of coalition formation games. Extensive simulation results demonstrate the practical applicability of the proposed approaches in real-time multi-robot applications.

As explained at the introduction to the thesis, the experimental evaluation has been done on a team of five robots that were designed and built in the laboratory. Considerable effort has been spent for this endeavor. Furthermore, the operation of these robots has necessitated the design and development of a novel control architecture. In a collaborative effort, a novel sense-communicate-act architecture has been developed and implemented in ROS. Finally, the problem of getting a set of robots in a loosely specified formation - referred to as realization problem - has been studied. In this study, initially an approach based on genetic algorithms has been developed. Due to computational complexity that hinders real-time applications, an alternative approach based on artificial potential functions has also proposed as presented. However, further study is required in order to study its robustness properties.

APPENDIX A: MULTI-ROBOT DYNAMICS FOR NAVIGATION

Each robot's dynamics obeys a differential equation as defined by the gradient flow of an appropriately defined function $\varphi_i : \mathcal{F} \times \mathcal{G} \to [0, 1]$ that encodes its task as:

$$\dot{b}_i = -D_{b_i}\varphi_i(b, \chi_i(g)) \quad \forall i \in \mathcal{R} \tag{A.1}$$

Of course, more general formulations are also possible. The information set $\chi_i(g) \subset \mathcal{R}$ of each robot is a set-valued function that specifies the set of robots whose configuration states are available to it [120]. We assume that each robot i is endowed with sensors that allow it to know its own state b_i while those of other robots can only be acquired via inter-robot link. If $\chi_i(g) = \{i\}$, it only knows about itself and is completely oblivious to other robots. If the information set consists of itself and immediate neighbors as $\chi_i(g) = \{i\} \cup \mathcal{N}_i(g)$, then the robot gets configuration state information from neighboring robots. In this case, as network changes, so does each information set $\chi_i(g)$. If the cardinality of this set is $|\chi_i(g)| = r$, then the network is all-to-all. If $|\chi_i(g)| << r$, then the robot has a direct link with only a subset of robots. If this holds for all the robots, then network density will be considerably lower compared to all-to-all network.

We use a feedback based approach that is a modified version of artificial potential functions [104, 121]. In this framework, if the robots are not communicating ($\chi_i(g) = \{i\}$) at all and hence are totally myopic, the probability of successful task termination is expected to be low. On the other hand, with complete information ($\chi_i(g) = \mathcal{R}$), almost all tasks can be completed as presented in [121]. Different from that work, with pairwise game theoretic communication, each robot exchanges information only with robots in $\chi_i(g) = \{i\} \cup \mathcal{N}_i(g)$. In the sequel, we present a summary of this approach for completeness. The interested reader is kindly referred to [121] for details. First define

the function $\widehat{\varphi}_i : \mathcal{F} \times \mathcal{G} \to [0, \infty)$ that encodes the task.

$$\widehat{\varphi}_i(b, \chi_i(g)) = \frac{\gamma_i(b, \chi_i(g))^{k_i}}{\beta_i(b, \chi_i(g))}$$

It consists of two terms. The function γ_i encodes proximity to goal positions not only for itself, but also those of its immediate neighbors with which it is in communication with as:

$$\gamma_i(b, \chi_i(g)) \quad = \sum_{j \in \chi_i(g)} (b_j - h_j)^T (b_j - h_j)$$

Hence, from the perspective of each robot, as it is minimized iff both the robot and its neighbors reach their goal positions, information sharing enables cooperation among neighboring robots. This is required in order to ensure that in cases where robots block each other (i.e. one robot being at its goal position blocking other robots reaching their destinations), the blocking robots move out of their goal positions in order to allow the blocked robots to do so. The denominator β_i encodes the distance from freespace boundary and thus information exchange here is used for collision avoidance among neighboring robots.

$$\beta_i(b, \chi_i(g)) = \beta_{0i} \prod_{\substack{j \in \chi_i(g) \\ j \neq i}} \beta_{ij} \beta_{0j}$$

Here $\beta_{ij} = (b_i - b_j)^T (b_i - b_j) - \rho_{ij}^2$ denotes the pairwise distance and $\beta_{0j} = \rho_{0j}^2 - \|b_j\|^2$ denotes the proximity to the workspace boundary. The parameter $k_i \in \mathbb{Z}^+$ is a relative weighting parameter. The function $\widehat{\varphi}_i$ is made admissible via composing it with $\sigma : [0, \infty) \to [0, 1]$ defined as by $\sigma(x) = \frac{x}{x+1}$. In order to make the goal a non-degenerate critical point, further composition with a sharpening function $\sigma_d : [0, 1] \to [0, 1]$ defined as $\sigma_d(x) = x^{1/k_i}$ is used. Hence, the function φ_i is defined as:

$$\forall i \in \mathcal{R} \quad \varphi_i(b, \chi_i(g)) = \sigma_d \circ \sigma \circ \widehat{\varphi}_i(b, \chi_i(g))$$

The solution of this family of differential equations describes the state trajectory of each robot. The robots continue moving until $\forall i \in \mathcal{R}$, $\dot{b}_i = 0$ which in turn implies that at the corresponding configuration state b^*, $\forall i \in \mathcal{R}$, $D_{b_i}\varphi_i(b^*, \chi_i(g)) = 0$. Of course, whether the associated task is achieved or not depends on the task specific properties of b^*.

APPENDIX B: TASK HANDLING ALGORITHM

The task handling automaton in Figure 4.1 is defined by the algorithm as shown in Figure B.1 and Figure B.2.

1: $t \leftarrow 0$ ▷ Initialize time
2: $k \leftarrow 0$ ▷ Initialize task index
3: $w \leftarrow 0$ ▷ Initialize temporary index for task
4: *state* $\leftarrow$ "idle" ▷ Initialize state of handling process
5: **while** in operation **do**
6: **if** new task **then**
7: $k \leftarrow k + 1$
8: $h^i_{k,1} \leftarrow t$ ▷ Note task time
9: $h^i_{k,2} \leftarrow 0$
10: $\mathcal{H}^i \leftarrow \mathcal{H}^i \cup h^i_k$
11: **if** *state* is "idle" **then**
12: Ask for assistance from the neighbors $\mathcal{N}_i(g)$
13: **if** $\exists$ available assisting robot $j \in \mathcal{N}_i(g)$ **then**
14: $t_s \leftarrow t$
15: $h^i_{k,2} \leftarrow 1^-$ ▷ Status to "being handled"
16: *state* $\leftarrow$ "handling"
17: **else**
18: $w \leftarrow k$ ▷ waiting for assistance
19: *state* $\leftarrow$ "waiting"
20: **end if**
21: **end if**
22: **end if**

Figure B.1. Algorithm for handling tasks of robot i.

if $\exists\ h_l^i \mid h_{l,2}^i \neq 1^- \ \&\ t - h_{l,1}^i \geq \Delta t_o$ **then**
  $h_{l,2}^i \leftarrow -1$ ▷ Status to "not handled"
end if
if $\exists\ h_l^i \mid h_{l,2}^i = 1^- \ \&\ t - t_s \geq \Delta t_h$ **then**
  $h_{l,2}^i \leftarrow 1$ ▷ Status to "handled"
  state $\leftarrow$ "idle"
end if
if (*state* is "idle" $\&\ \exists\ h_l^i \mid h_{l,2}^i = 0 \ \&\ l \neq w$) or (*state* is "waiting" $\&\ \exists\ h_l^i \mid l = w$) **then**
  if $\exists$ available assisting robot $j \in \mathcal{N}_i(g)$ **then**
    $t_s \leftarrow t$
    $h_{l,2}^i \leftarrow 1^-$ ▷ Status to "being handled"
    state $\leftarrow$ "handling"
  else
    $w \leftarrow l$ ▷ waiting for assistance from $\mathcal{N}_i(g)$
    state $\leftarrow$ "waiting"
  end if
end if
if request assistance from robot $j \in \mathcal{N}_i(g)$ **then**
  if *state* is "idle" **then**
    state $\leftarrow$ "assisting" ▷ Start assisting robot j
    $t_s \leftarrow$ t
  else
    Reject the request
  end if
end if
if *state* is "assisting" $\&\ t - t_s \geq \Delta t_h$ **then**
  state $\leftarrow$ "idle"
end if
$t \leftarrow t + \Delta t$
end while

Figure B.2. Algorithm for handling tasks of robot i (continued).

APPENDIX C: ROBOTIC PLATFORM DEVELOPMENTS

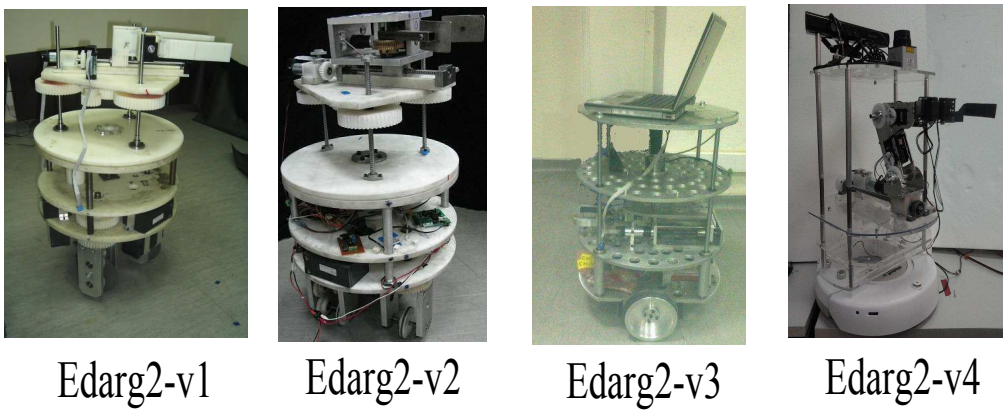


Figure C.1. The evolution of mobile robot platforms.

The four different robots we have designed and produced from the beginning of the thesis are shown in Figure C.1. The history of these robots is as follows:

(i) EDARG2-v1: The EDAR G2 robot (Event Driven Assembler Robot 2nd Generation) has been redesigned and built in the Intelligent Systems Laboratory. The robot is as shown in Figure C.1. It consists of a two degrees of freedom base, a three freedom arm and a one degree of freedom gripper. Electrical infrastructure and software framework has been developing. EDARG2 has capabilities of grasping and holding parts. The tests performed on the robot revealed multiple problems with the mechanical design and realization. Major problems are as follows:

- Translational and rotational motors' shafts are not properly mounted to the gear. Hence set-screw damages motors' shaft.
- Misalignment in mounting gears.
- Balance problem while moving.

Solutions to the problems:

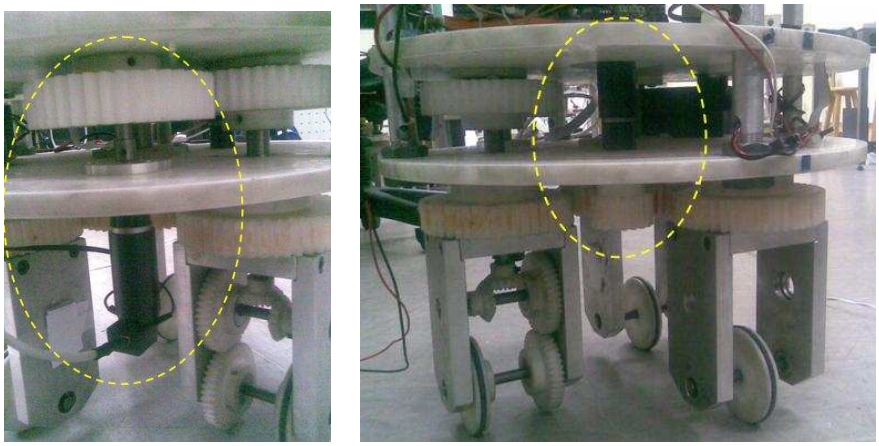
- Gears should be produced according with motor's shaft shape.
- Positions of the gears on the planes should be processed simultaneously.
- Coupling mechanism should be used in the translational motor since it needs

more torques than the others.

- For balance of the robot, two wheels can be used instead of one wheel.

(ii) EDARG2-v2: The robots need to have smoothly running mechanical systems in order to be fully controllable and operational. Substantial testing of the robot units at the beginning of this term allowed us to fully test and diagnose the mechanical abilities. These tests showed that certain mechanical problems that would be detrimental to the robots' operation and full functioning had not been solved. These problems could cause damage to the electromechanical parts such as overloading of the motors. These problems and solutions can be summarized under three categories as follows:

- Driving the gears directly: As shown in Figure C.2, linear and rotational motors directly drive their gear. Due to the direct driving, misalignment of the motors or the gears forces the shaft, thereby damaging the motors. A solution to this problem is to design and manufacture a new coupling mechanism. This was done for the translational and the rotational motors.



(a) (b)

Figure C.2. (a) Linear motor and its gear group; (b) Rotational motor and its gear group.

- Problems with the aluminum beams: Each robot has four aluminum beams connecting parts of its structure as shown in Figure C.3. As these beams were not identical and were not symmetrically positioned, motors and gears had positioning problems. The solution to this problem was to manufacture

a new aluminum beams and re-mount them.

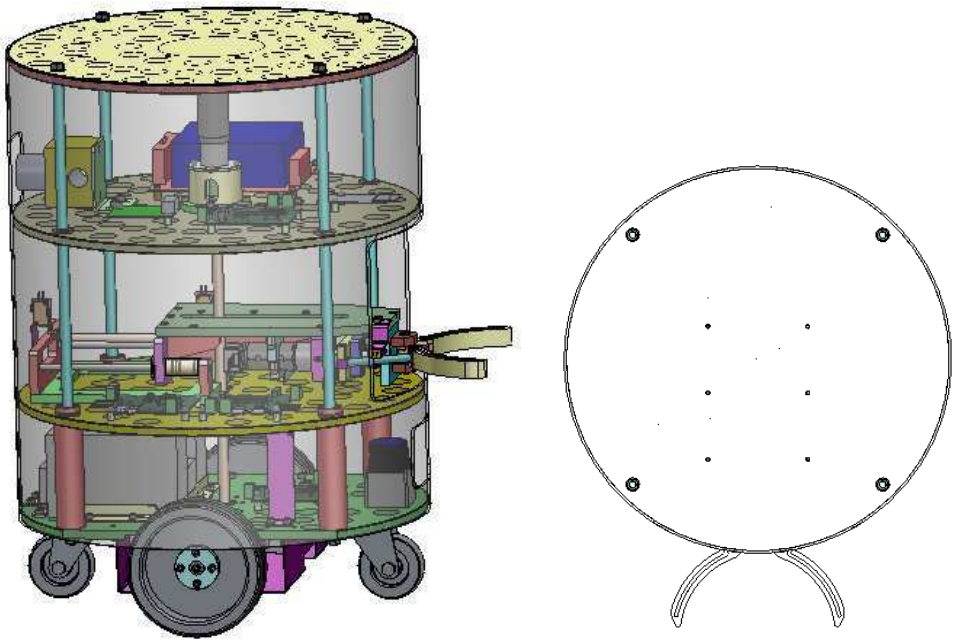
Figure C.3. Aluminum pipes connecting the planes.

- Setscrews: They are used to fasten motor's shaft to a gear as shown in Figure C.4. Under overload, these screws may slip off the motor's shaft. The solution to this problem was to erode the inside of the parts being connected to motor shafts based on the shape of the corresponding motor shaft.

Figure C.4. Setscrews.

(iii) EDARG2-v3: New design for EDARG2 shown in Figure C.5 is based on the differential drive which is the simplest possible drive mechanism. This mechanism consists of two wheels controlled by separate motors. The newly designed robot has also capabilities of grasping and holding parts. After manufacturing the first prototype, some electro-mechanical tests were applied to detect the efficiency of the mechanical units. The detected problems are as follows:

- Left driving mechanism is stressed more than the right mechanism. Therefore, the robot cannot move straight.
- Gripper mechanism cannot move forward.
- The distance the gripper gets away from the body is too small that it cannot approach to any part without colliding with it.

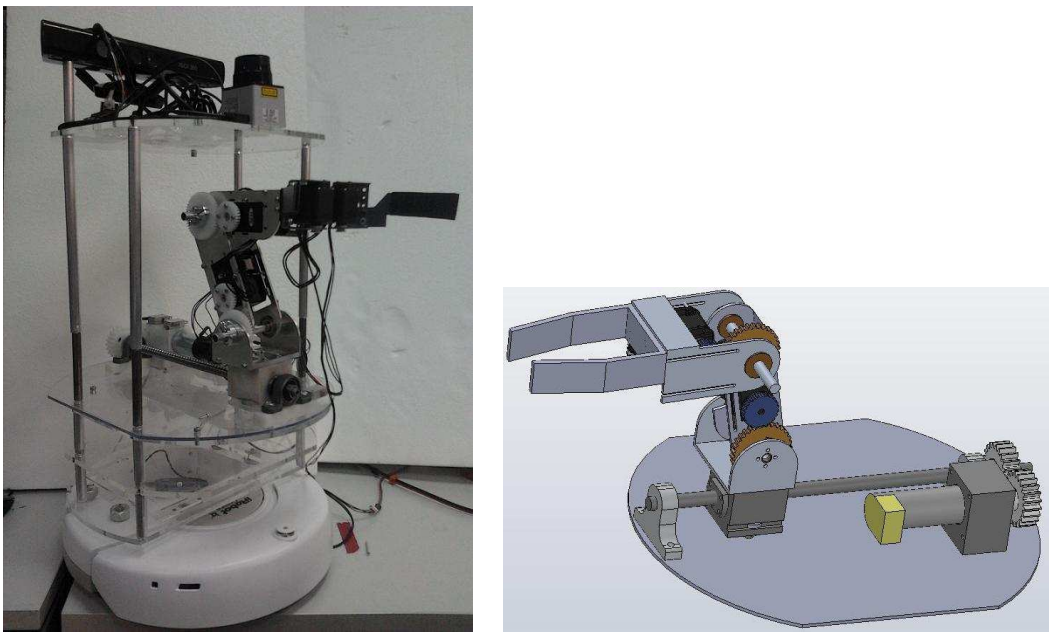


(a) (b)

Figure C.5. (a) New design for EDARG2; (b) Top view of the design.

(iv) EDARG2-v4: This new design was based on the differential drive which is the simplest possible drive mechanism. However, while the project was about to be completed, the firm with which we have made a contract canceled the project due to their technical inadequacy. Hence, we plan to buy mobile robot platform and design and manufacture the gripper mechanism for this platform.

The best option for the mobile robot platform is IRobot Create shown in Figure C.6a since it is the cheapest mobile robot base and is being used widely in the robotic community. The hardest part in the project is to design and manufacture light and small gripping mechanism. For this purpose, we worked with a group of students from Department of Mechanical Engineering. The designed griper is shown in Figure C.6b. The gripper will be placed between the top two planes in



(a) (b)

Figure C.6. (a) New platform for EDARG2; (b) The designed gripper.

the robot.

APPENDIX D: EDARG2 Robot

EDARG2 is a differential wheel autonomous robot with a development history as explained in Appendix C. The last generation of this robot is as shown in Figure D.1. The robot can be used both manually by connecting remotely or in automated mode.

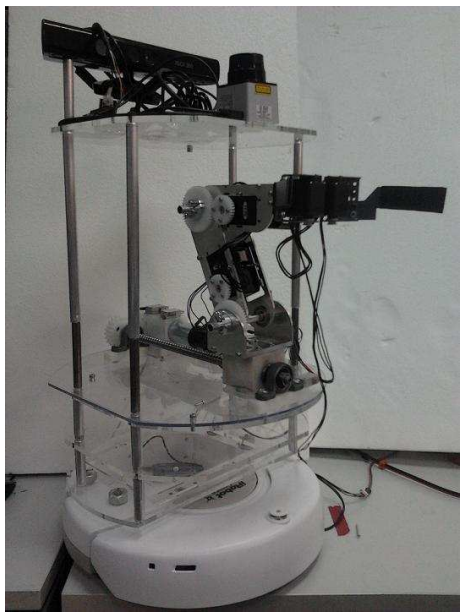


Figure D.1. EDARG2.

D.1. Hardware

The technical specifications of the robots are as follows:

- Driving Mechanism: Differential drive
- Diameter: 33.5cm
- Height: 60cm
- Maximum speed: 0.5 m/s
- Weight: 6 kg
- Wheel Diameter: 63.5 mm
- Wheel Encoders: 360/32 degree, 2 cm accuracy

- Maximum Moving Time: 3.5 hours (with 3Ah battery fully charged and no payload)
- Communication with Robot Base: UART
- Battery for Hokuyo and Kinect: 12V 4.5AH
- DC to DC Converter for Hokuyo: Converting 12V to 5V

Its processing is done by a Asus Eee PC Netbook (Dual Core 1.6GHz Processor, 2 GB DDR3, 320GB HDD and WLAN 802.11 b/g/n@2.4GHz). Furthermore, its sensing capabilities are as follows:

- Kinect (laser range scanner + camera),
- Hokuyo URG-04LX,
- Cliff sensors,
- Bump sensor

The wiring diagram of EDARG2 can be seen in Figure D.2.

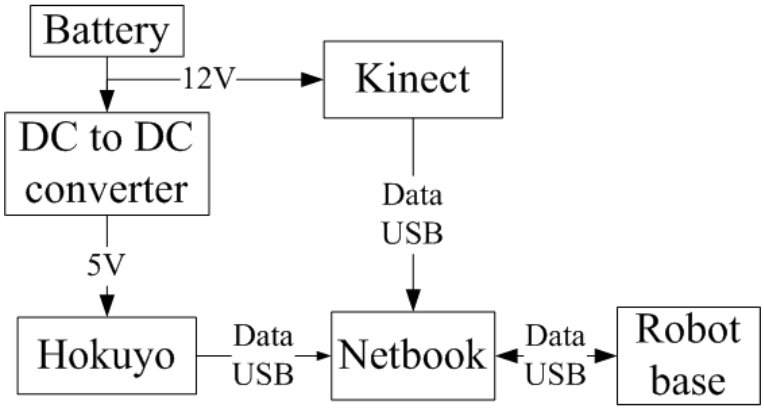


Figure D.2. Wiring diagram of EDARG2.

D.2. Software

The software of the robots requires the installation of the following software:

- Operating System: Ubuntu 12.04 LTS
- ROS (Robot Operating System) Fuerte, Desktop Sharing, Remote Desktop Viewer
- Additional Libraries: MPFR-v3.1.2, GNU Scientific Library-v1.15
- Software Development Environment: Qt-SDK-Linux-x86-v1.2.1

D.3. User's Guide

Before starting, the wiring should be made appropriately according to the wiring diagram as shown in Figure D.2.

D.3.1. Powering the Robots

While the robot base and computing unit have their own integrated batteries, Hokuyo and Kinect sensors require 5V and 12V power sources, respectively. The external 12V source should be converted into 5V using DC to DC converter for Hokuyo sensor. After powering on the robot base, the netbook and the sensors, the robot is ready for the operation.

D.3.2. Remote Connection

The robot can be controlled via remote connection. The following steps need to be followed:

- Desktop Sharing program should be working on the robot. It should also be noted that the robot should be connected to the network (either via WiFi or LAN connection).
- The remote computer connects to the robot via Remote Desktop Viewer program which is on the same network with the robot.
- The robot can be teleoperated with a keyboard using 'turtlebot_teleop' package in ROS.

D.3.3. Automated Mode

The following steps should be done sequentially in order to run the robots in automated mode.

- Connect to the current ad hoc network.
- Run roscore to activate ROS.
- The coordinator robot runs the communication node:
  `rosrun communicationISL communication`
- The coordinator robot runs the following launch file
  `experiment-irobot-coor.launch`:

```
<launch>
<node name="map_server" pkg="map_server" type="map_server"
args="/home/irobot3/fuerte_workspace/sandbox/map/haritaYagmur.yaml"/>
<node name="hokuyo_node" pkg="hokuyo_node" type="hokuyo_node">
<param name="frame_id" value="/base_laser"/>
</node>
<node name="tf_broadcaster" pkg="robot_setup_tf" type="tf_broadcaster"
output="screen" />
<node pkg="tf" type="static_transform_publisher" name="link1_broadcaster"
args="0 0 0 0 0 0 base_footprint base_link 100" />
<node name="turtlebot_node" type="turtlebot_node.py" pkg="turtlebot_node">
    <rosparam>
      publish_tf: True
      has_gyro: True
      odom_angular_scale_correction: 1.3043
      odom_linear_scale_correction: 1.0
      gyro_scale_correction: 1.4556
      gyro_measurement_range: 250.0
    </rosparam>
  </node>
<node name="amcl" type="amcl" pkg="amcl">
    <rosparam>
      odom_alpha1: 0.1
      odom_alpha2: 0.1
      odom_alpha3: 0.1
```

```
      odom_alpha4: 0.1
    </rosparam>
  </node>
<node name="navigation" pkg="navigationISL"
type="navigation" output="screen" />
<node name="hotspothandler" pkg="hotspothandlerISL"
type="hotspothandler" output="screen" />
<node name="hotspotobserver" pkg="hotspotobserverISL"
type="hotspotobserver" output="screen" />
</launch>
```

- Regular robots run the following launch file

  `experiment-irobot.launch`:

```
<launch>
<node name="map_server" pkg="map_server" type="map_server"
args="/home/irobot4/fuerte_workspace/sandbox/map/haritaYagmur.yaml"/>
<node name="hokuyo_node" pkg="hokuyo_node" type="hokuyo_node">
<param name="frame_id" value="/base_laser"/>
</node>
<node name="tf_broadcaster" pkg="robot_setup_tf"
type="tf_broadcaster" output="screen"/>
<node pkg="tf" type="static_transform_publisher" name="link1_broadcaster"
args="0 0 0 0 0 0 base_footprint base_link 100" />
<node name="turtlebot_node" type="turtlebot_node.py" pkg="turtlebot_node">
    <rosparam>
      publish_tf: True
      has_gyro: True
      odom_angular_scale_correction: 1.3043
      odom_linear_scale_correction: 1.0
      gyro_scale_correction: 1.4556
      gyro_measurement_range: 250.0
    </rosparam>
  </node>
<node name="amcl" type="amcl" pkg="amcl">
    <rosparam>
      odom_alpha1: 0.1
      odom_alpha2: 0.1
      odom_alpha3: 0.1
```

```
      odom_alpha4: 0.1
    </rosparam>
  </node>
<node name="communication" pkg="communicationISL"
type="communication" output="screen" />
<node name="navigation" pkg="navigationISL"
type="navigation" output="screen" />
<node name="hotspothandler" pkg="hotspothandlerISL"
type="hotspothandler" output="screen" />
<node name="hotspotobserver" pkg="hotspotobserverISL"
type="hotspotobserver" output="screen" />
</launch>
```

- The coorinator robot runs the coordinator node:

  ```
  rosrun coordinatorISL coordinator
  ```

  The robots use the following configuration file based on JSON format:

configISL.json:

```
"linkCost":"1.5",
"maxGameMove":"6000",
"checkPSPeriod":"10",
"numrobots":"4",
"Tg":"15",
"Tc":"4",
"radius":"17.5",
"linearVelocity":"0.1",
"angularVelocity":"0.4",
"angleThreshold":"20",
"distanceThreshold":"20",
"ro":"300",
"kMin":"8",
"kMax":"10",
"partDist":"300",
"iscoordinator":"1",
"robotID":"3",
```

```
"initialX":"0",
"initialY":"180",
"targetX":"120",
"targetY":"210",
"poissonMU":"0.1",
"timeoutHotspot":"20",
"handlingDuration":"10",
"waitingDuration":"2",
"amclfakeLoopRate":"0.5",
"amclfakeIncrement":"2",
"Robots":{"Robot":[{"name":"IRobot1","ip":"10.42.0.1","coordinator":"0",
"initialX":"-180","initialY":"-90"},{"name":"IRobot2","ip":"10.42.0.2",
"coordinator":"0", "initialX":"30", "initialY":"30"},{"name":"IRobot4",
"ip":"10.42.0.4","coordinator":"0", "initialX":"-150", "initialY":"210"},
{"name":"IRobot5", "ip":"10.42.0.5","coordinator":"0", "initialX":"-240",
"initialY":"0"}]}, "Obstacles":{"Obstacle": [{"id":"1","radius":"45",
"x":"-100","y":"25"}, {"id":"2","radius":"45","x":"180","y":"120"}, {"id":"3",
"radius":"45", "x":"130","y":"-120"}]}
```

APPENDIX E: CONTROL ARCHITECTURE

In this section, the control architecture that was developed for multi-robot systems is described. This architecture extends the sense-act loop of control architectures to sense-communicate-act paradigm. In this new paradigm, the control architecture has to incorporate two new modules: communication and network topology update. The communication module enables the exchange of information with neighboring robots as deemed by the current communication network. As the robots and environment are both dynamic, the communication network topology has to be evolving dynamically. In network topology update module, the topology is updated periodically considering the communication or task related objectives. There are two alternative approaches – centralized or decentralized network update. The two modules are integrated into ROS. With this extended architecture, multi-robot scenarios with network based communication can easily be developed. This is a collaborative work done together with Hakan Karaoguz. The proposed architecture is developed in the framework of Robot Operating System (ROS) - an open-source robot meta-operating system. With this extended architecture, multi-robot application scenarios with network based communication can easily be developed. The rest of this section summarizes the system as presented in [103]. The robot architecture for a multi-robot system is presented in Section E.1. The communication module is explained in Section E.2. In Section E.3, the network update module is described.

E.1. Control Architecture

The control architecture is comprised of three layers [122]. The physical layer consists of actuator and sensors modules. The actuator module consists of motors and their drivers. The sensors read the internal state of the robot as well as providing sensory feedback. This layer is hardware specific and needs to be modified as robotic platform or components are changed.

The next layer is the hardware abstraction layer. This layer serves as an inter-

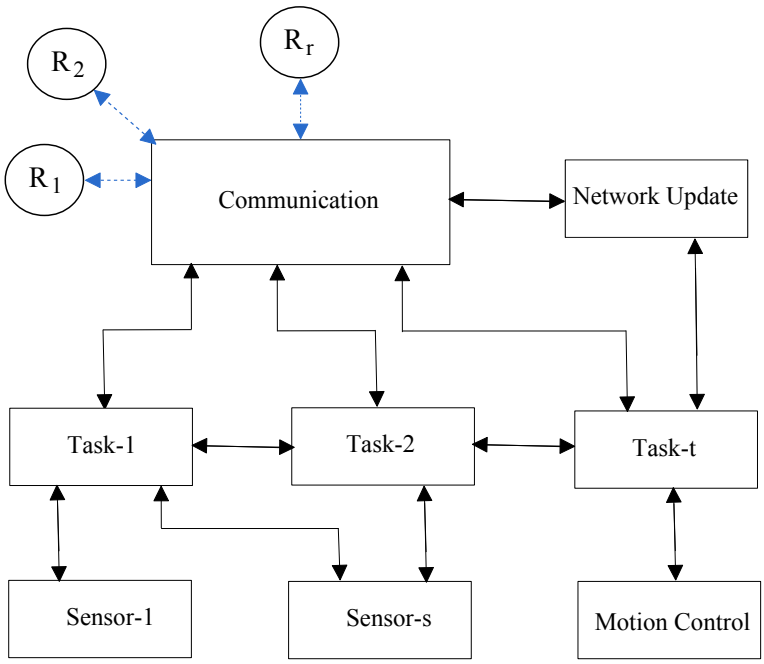


Figure E.1. Functional layer. Rectangles represent the nodes in the ROS. Circles show the other robots.

mediate layer between the physical layer and robotic modules. They are independent of the robotic platform or the hardware components. We use ROS which provides hardware abstraction, device drivers, message-passing and package management. In particular, the following properties are extremely convenient for multirobot applications [123]:

(i) Modularity: Separate processes (navigation, network update, mapping) can all be separated out. They interact through an interface in which software processes (a.k.a. "nodes" in ROS) communicate about shared "topics" in ROS. Publish/Subscribe allows each node to send and receive only the desired data (messages).

(ii) Separation of physical and messaging interface helps avoid hardware dependencies.

(iii) Inter-module communication is enabled via asynchronous callback functions that are called whenever data is available for processing.

The final layer is the functional layer. It is comprised of a collection of robotic modules.

(i) Motion control: This module updates the velocity of the wheels and providing the odometry information.

(ii) Sensing: These modules are responsible for providing sensory information to the other modules from the sensors. A separate module is associated with each different sensor.

(iii) Communication: These modules are associated with inter-robot communication.
- Communication: This module is responsible for communication with other robots via sending or receiving messages.
- Network update: This module updates the communication state a_i.

(iv) Task modules: These modules associated with the different capabilities and functionalities of robots. Some of these modules are essential – meaning that all the robots will have these modules. Other modules are optional and the robot may not have them at all.
- Navigation: This module generates the velocity profile for moving the robot in its workspace and thus affects b_i. It sends the current b_i to the communication module every T_c seconds.
- Localization: This module localizes the robot and generates b_i for each robot individually.

Each module is active concurrently with the other modules and may interact possibly with many of them. These modules are implemented as "nodes" in the ROS as shown in Figure E.1. Each node behaves like a separate process, but can interact with the other nodes via the publish/subscribe mechanism provided by the ROS. As the robots are engaged in a particular task, some of the task modules will be activated depending on what's required of the robot.

E.2. Communication Module

The communication module of robot i is responsible for the incoming and outgoing messages. A TCP based communication protocol is used. This module performs

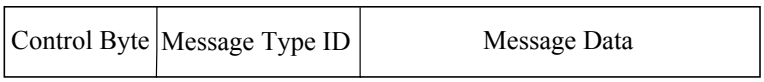


Figure E.2. Message structure.

the following:

- Open a socket with each robot $j \neq i$ in the team.
- Receive the incoming messages from the sender robots.
- After decoding the message, publish the incoming message to the related node(s).
- After receiving the outward bounded messages from the nodes, encode the outgoing message and send it to the receiver robot.

Note that all the sockets are opened at the beginning of operation in order not to make an extra effort on opening and closing sockets as the network topology changes. However, only some of these sockets will be in use depending on the network topology. The remaining sockets will be idle. Let it be noted that having opened, but idle sockets requires only negligible computation and communication overhead. This is because both end-points generate and respond to periodic TCP-generated keep-alive messages that occur at least once every 2 hours [124].

The message format is based on the modified version of CVS (comma-separated values). Each message contains three parts as shown in Figure E.2.

- Control Byte: This byte is for error checking purposes and is set to "AA".
- Message Type ID: This byte specifies the type of information in the message.
- Data: This part is of variable length as depending on the message type.

Note that communication is done in a robot-to-robot manner instead of broadcast. Hence the message data does not have any robot identity information regarding the sender robot.

E.3. Network Update Module

Hence, network update is based on modifying the collective communication state a based on either decentralized or centralized network topologies.

E.3.1. Decentralized Network Update Strategy

In this approach, each robot decides for its neighbors $\mathcal{N}_i(g)$ independently of other robots by updating its communication state a_i every T_g seconds. The network update proceeds as follows:

- This module is activated every T_g seconds.
- It attempts a series of temporary direct communications with other robots and updates a_i directly.
- The update process is continued until a predefined time ΔT_g is reached.

As robots are resource constrained, each robot is viewed as maximizing its own benefit from participation in the communication network. Thus, the updating of the network topology is based on an payoff function $v_i : \mathcal{G} \times \mathcal{F} \rightarrow \mathbb{R}$. For details, the reader is referred to Chapter 2.

E.3.2. Centralized Network Topology

In the centralized approach, a network coordinator is responsible for determining the network topology. Hence, the functionality of network update module depends on whether the particular robot is a regular robot or a network coordinator. In case of a regular robot, the network module simply sends position information b_i to the coordinator and receives the updated communication state a_i as:

- Send position state b_i to the network coordinator periodically every T_g seconds.
- Update the collective communication state a_i based on received information from the network coordinator.

In turn, the coordinator does the following:

- Receive position b_i from all the robots.
- Find a network topology acceptable to all the robots based on received information and update a. It ensures that sent information is consistent - namely $a_{ij} = a_{ji}$.
- Send a_i to each robot.

The topology update is based on robots' communication objectives $v_i : \mathcal{G} \times \mathcal{F} \to \mathbb{R}$ that encode their communication strategies while they are engaged in their tasks. The set $\mathcal{F}$ refers to the set of all robots positions.

E.3.3. Inter-Robot Messages

The list of possible messages depends on the requirement of the application. In our case, we define three types of message in the implementation.

(i) Message Type ID = 1. In this case, the data contains robot's position state b_i in the global coordinate system. A sample message for this type is ‘‘AA’’, ‘‘1’’, ‘‘0;0’’ which says that my current position is at (0,0).

(ii) Message Type ID = 2. In this case, the data contains detailed robot information including the robot's current position, radius, and goal position. A sample message for this type is ‘‘‘AA’’, ‘‘2’’, ‘‘0;0;17.5;500;500’’, which indicates that I am a robot with 17.5 cm radius at (0,0) location and my desired position is at (500,500).

(iii) Message Type ID = 3. This message is sent from the coordinator to a robot. The message data contains a list of $\mathcal{N}_i(g)$ of each robot. A sample message for this type is ‘‘AA’’, ‘‘3’’, ‘‘1; 2’’, which indicates that Robot1 and Robot2 are your neighbors. If there is no neighbors, the message would become ‘‘AA’’, ‘‘3’’, ‘‘0’’.

APPENDIX F: PLANAR MULTI-ROBOT REALIZATION USING GENETIC ALGORITHM

This chapter considers the problem of planar multi-robot realizations of connectivity graphs. The problem is defined as the generation of robot planar positions whose connectivity graph is identical to an *a priori* given connectivity graph with the additional constraint that the robots must not be overlapping with each other. This problem arises in many applications such as exploration, search, patrolling and collective games (such as soccer) that require automatic positioning of multiple robots with a particular underlying connectivity graph constraint. For example, in multi-robot deployment and coordination tasks with limited communication, robot positions must satisfy the particular connectivity graphs [33, 125]. This chapter presents a stochastic approach to this problem and proposes a graph-based genetic algorithm for generating planar multi-robot realizations.

F.1. Related Literature

There are three related areas: robot networks, disk graphs, and graph drawing. In multi-robot systems, the concept of connectivity graphs has been introduced which imposes various constraints on the relative positions of the robots [126,127]. For example, connectivity graphs provide a graph-theoretic model for broadcast networks where the radii of the circles correspond to the communication range ρ_c. Interestingly, while many approaches are based on graph based models, the issue of whether an arbitrary graph has a multi-robot realization or not has been mostly overlooked.

Connectivity graphs are known as unit disk graphs in graph theory – which are the intersection graphs of closed disks in the plane where each vertex corresponds to a circle and edge appears between two vertices when the corresponding circles intersect [128]. Of course, the distance unit is not critical since the disks realize the same graph even if the coordinate system is scaled appropriately. Furthermore, unit disk graphs have

several alternative definitions that are all equivalent to each other up to a choice of scale factor. Two such alternatives are the intersection graph of equal radius circles or a graph formed from a collection of circles all having the same radius where two circles are connected by an edge if each circle contains the center of the other circle. The set of disks is said to realize the graph [129]. A realization is therefore a mapping of the vertices to points which realize the graph. The recognition problem of unit disk graphs is then posed as: Given a graph, determine if it has a realization [129]. It has been shown that recognizing unit disk graphs is NP-hard. The results are also shown to hold for the disk touching graphs – namely all disks have disjoint interiors.

The NP-hardness of the problem has motivated the development of approximate, potential-based or stochastic approaches – as many problems from different problem areas nevertheless require solutions to the planar realization problem. An inapproximability result has placed a bound on how well the planar coordinates can be derived from the connectivity information alone [130]. In graph drawing, where the goal is to produce aesthetically pleasing drawings of general undirected graphs, one proposed approach is the spring model algorithm. Here, the graph is viewed to be a mechanical collection of rings (the vertices) and connecting springs (the edges) with minimal energy configuration attained when the network graph approaches the goal graph [131]. However, as the spring method is likely to be trapped by local optima, the configurations that are obtained are very poor. In the genetic algorithm TimGA, aesthetic criteria used such as the number of edge crossings, even distribution of nodes, and edge length deviation are utilized [132]. An extension of this work that develops new mutation operators has been proposed in [133]. However, none of these algorithms enforce adherence to a given connectivity graph by considering both edges and non edges (namely vertex pairs that should not have a link between them). The novelty of the proposed approach is to consider the graph realization problem and to propose a genetic algorithm which has proved to be statistically working.

F.2. Connectivity Graphs

Suppose that the robots have limited communication range ρ_c. A disk-neighborhood of robot i is a closed ball $B_{\rho_c}(b_i)$ of radius $\rho_c >> \rho_i$ around $b_i \in \mathbb{R}^2$. Given ρ_c, any configuration $b \in \mathcal{F}$ induces a state-dependent mapping $g : \mathcal{F} \to \mathcal{G}$. Here $\mathcal{G} = \{g' | g' \subseteq K_p\}$ is the set of all possible graphs on $\mathcal{R}$ and K_r is the complete graph [24]. The image of the graph map is $g(b) = (\mathcal{R}, \mathcal{E}(b))$ is known as the connectivity graph [126]. Here $\mathcal{E}(b)$ is the set of edges as defined by the connectivity matrix $A(b) = [a_{ij}(b)]$:

$$\mathcal{E}(b) = \{ij \mid a_{ij} = 1\} \tag{F.1}$$

The connectivity matrix $A(b)$ is defined as follows:

$$a_{ij}(b) = \begin{cases} 1 & \delta_{ij} \leq \rho_c \text{ and } i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{F.2}$$

F.3. Problem Statement

Consider a graph function $g : \mathcal{F} \to \mathcal{G}$. Suppose we are given $g^* \in \mathcal{G}$. The set $g^{-1}(g^*) \subset \mathcal{F}$ represents the set of robot configurations all having graph g^*. The goal is to find a realization $b \in \mathcal{F}$ such that $b \in g^{-1}(g^*)$.

F.4. General Approach

Our approach is based on genetic algorithms. This Section first gives a broad overview of genetic algorithms. Following, we describe its adaptation to the planar realization problem.

F.4.1. Genetic Algorithms

Genetic algorithms are a class of evolutionary methods for determining the optimal classifiers or equivalently an optimal solution [134, 135]. In genetic algorithms, a classifier is represented by a string of genes that is also known as a chromosome. The mapping from a chromosome to the features of a classifier is flexible and depends on the application. In broad overview, genetic algorithms employ stochastic search to evolve the best chromosome. First, a population set of classifiers is constructed. Here each chromosome differs somewhat from the others in the population. Next, a fitness function that evaluates the goodness of each chromosome is constructed. This function is used to compute the score of each chromosome. Following, the classifiers are ranked according to their score and only the fittest are retained. These are then stochastically altered to generate the next generation. There are three primary genetic operators that govern reproduction: replication, crossover and mutation. Replication is mere reproduction. Crossover involves the mating of two different chromosomes via exchanging certain parts. Mutation occurs when the genes change. The overall process is repeated for the succeeding generation. The process is terminated when at least one chromosome has a score that exceeds an a priori specified value.

F.4.2. Adaptation to Planar Multi-Robot Realization

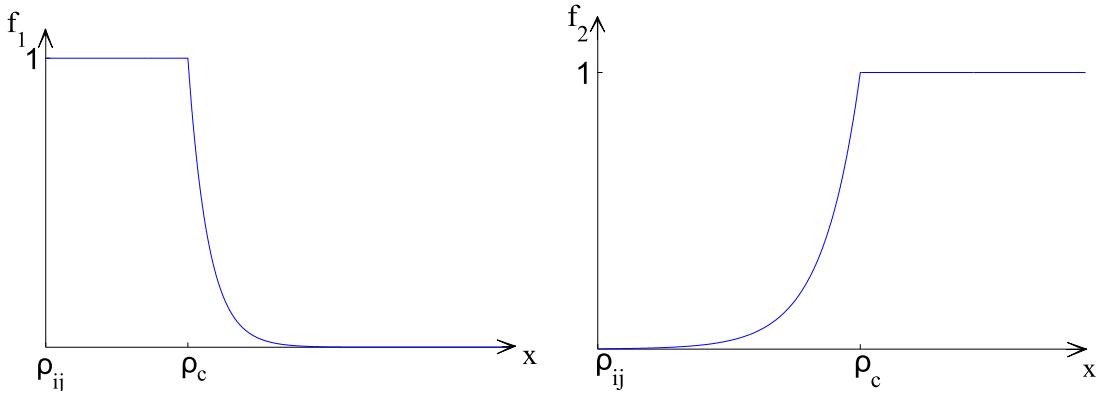
In employing genetic algorithms, we must first specify the map from a chromosome to the properties of the classifier. In the proposed approach, each chromosome corresponds to a particular state $b \in \mathcal{F}$ and hence a team of r robots that are all located within W. The goal is to generate a state (equivalently a chromosome) that has the given connectivity graph g^*.

An initial population set $S(0) \subset \mathcal{F}$ is constructed where the cardinality $|S(0)|$ is set a priori to N_P. Next, a selection process initiates a new generation $S(k) \subset \mathcal{F}$ where $k \in Z^+$ is the generation number. The fitness of each chromosome in the current population is evaluated based on a fitness function f and the members of this population set are ranked accordingly. The population $S(k)$ is via selecting members

from $S(k-1)$ randomly with probability depending on the relative rank value of the individuals [134]. In this manner, population members having higher fitness are chosen more than those having lower fitness values. These members of the population are to be used in the stochastic alteration that follows. Two primary genetic operators govern reproduction: crossover and mutation. This is followed by replacement where the population is changed via replacing the children with the parents. Elitism is applied to keep the best chromosome in new population. If there is no improvement in the elite chromosome after a predefined number of generations N_E, newly generated randomly realizations are used to replace the p_A percent of the population while keeping the best p_B percent of the population. The process is halted when fitness of a generation reaches a desired level or when the number of generations exceeds a given value N_G.

F.5. Fitness Function

Given a member $b \in S(k)$ of any population, the fitness function should measure the similarity between its graph $g(b)$ and the goal g^*. This is equivalent to finding $b \in \mathcal{F}$ in such a way that its adjacency matrix $A(b)$ is the same as that $A = [a_{ij}]$ of the given goal graph g^*. Recalling that $\delta_{ij} = \parallel b_i - b_j \parallel$, the fitness function $f : \mathcal{F} \rightarrow \mathbb{R}^{\geq 0}$ encodes the following measures.



(a) (b)

Figure F.1. (a) f_1 function; (b) f_2 function.

First, the similarity of adjacency matrix $A(b)$ of a given realization b with that

of the goal A is measured. For this, both the edges and no edges of $A(b)$ must be compared with the corresponding entities of A. If the goal graph has an edge ij and hence $a_{ij} = 1$, then $\rho_{ij} \leq \delta_{ij} \leq \rho_c$ which is measured by the function $f_1 : \mathbb{R}^{\geq 0} \rightarrow [0, 1]$ defined as:

$$f_1(x) = \begin{cases} 1 & x \leq \rho_c \\ e^{-(x-\rho_c)} & x > \rho_c \end{cases} \tag{F.3}$$

Similarly, if the goal graph does not have the edge ij, then $a_{ij} = 0$ which means that $\delta_{ij} > \rho_c$ which can be measured by the function $f_2 : \mathbb{R}^{\geq 0} \rightarrow [0, 1]$ defined as:

$$f_2(x) = \begin{cases} 1 & x > \rho_c \\ e^{(x-\rho_{ij})}/e^{(\rho_c-\rho_{ij})} & x \leq \rho_c \end{cases} \tag{F.4}$$

Hence, the term $\sum_{ij} a_{ij} f_1(\delta_{ij}) + (1 - a_{ij}) f_2(\delta_{ij})$ varies 0 in case of being completely different and $r(r-1)$ in case of being completely identical.

Next, we also measure the amount of dissimilarity between the adjacency matrix $A(b)$ of b and that A of the goal. It is composed of two terms. For all missing edges in $A(b)$ that are present in A, the function $f_3 : \mathbb{R}^{\geq 0} \rightarrow \mathbb{R}$ measures the distance each robot pair need to approach in order to come within each others' communication range ρ_c:

$$f_3(x) = x - \rho_c \tag{F.5}$$

Similarly, for all the superfluous edges present in $A(b)$ that should not be present with respect to A, the function $f_4 : \mathbb{R}^{\geq 0} \rightarrow \mathbb{R}$ measures how close is the corresponding robot pair from moving within each others' neighborhood as:

$$f_4(x) = 2\rho_o(1 - x/\rho_c) \tag{F.6}$$

The term $1 + \sum_{\substack{ij \notin \mathcal{E}(b) \\ ij \in g^*}} f_3(\delta_{ij}) + \sum_{\substack{ij \in \mathcal{E}(b) \\ ij \notin g^*}} f_4(\delta_{ij})$ varies between 1 when the target connec-

tivity graph is realized and being a large number in case of being different.

Hence, the fitness function can be constructed as the ratio of these two terms as:

$$f(b) = \frac{\sum_{ij} a_{ij} f_1(\delta_{ij}) + (1 - a_{ij}) f_2(\delta_{ij})}{1 + \sum_{\substack{ij \notin \mathcal{E}(b) \\ ij \in g^*}} f_3(\delta_{ij}) + \sum_{\substack{ij \in \mathcal{E}(b) \\ ij \notin g^*}} f_4(\delta_{ij})} \tag{F.7}$$

F.6. Reproduction

Two primary genetic operators govern reproduction: crossover and mutation.

F.6.1. Crossover

For crossover reproduction, first, a mating pool is generated via considering each robot $i \in \mathcal{R}$ and adding it to the mating pool with probability p_C. If the number of robots in this set turns out to be an odd number, a randomly selected robot is removed from the pool. The crossover operator considers each consecutive two members $b^-, b'^- \in S(k)$ of the mating pool and replaces them by two offsprings b^+ and b'^+. The superscripts $-$ and $+$ indicate each chromosome before and after the crossover respectively. Three types of crossover operators are considered: Single, multi-point and square.

F.6.1.1. Single Crossover. In the single point crossover, the location of a randomly selected robot i is swapped between the parents. That is to say, $b_i^+ = b_i'^-$ and $b_i'^+ = b_i^-$. All the links are adjusted accordingly. If the crossover leads to an unfeasible configuration for any of the resulting offsprings, the offsprings are not added to the population $S(k)$. Instead, the crossover is repeated with another robot $l \neq i$. If a feasible configuration is not available after trying all the indices, the crossover operation is not applied to this pair.

F.6.1.2. Multi Point Crossover. Multi-point crossover is similar to single point crossover. Here, the parents are spliced into two groups via choosing a robot index i randomly. The offsprings are generated via exchanging one of the groups formed. b^+ and b'^+ are defined as follows for $\forall j \in \mathcal{R}$:

$$b_j^+ = \begin{cases} b_j'^- & \text{if } 1 \leq j \leq i \\ b_j^- & \text{if } i < j \leq r \end{cases} \tag{F.8}$$

$$b_j'^+ = \begin{cases} b_j^- & \text{if } 1 \leq j \leq i \\ b_j'^- & \text{if } i < j \leq r \end{cases} \tag{F.9}$$

If the resulting operation leads to an unfeasible configuration for any of the offsprings, again the offsprings are not added to the population set $S(k)$. Instead, another robot index $l \neq i$ is selected. This is repeated until either both of the offsprings have feasible configurations or all the indices are depleted. In case of failure to generate offsprings having feasible configurations, the pointwise crossover is not done for this pair.

F.6.1.3. Square Crossover. In square crossover, two offsprings are generated via exchanging a small set of robots between the two parents [132]. The exchange is based on two square regions $C^-, C'^- \subset W$ both centered at the same location $c^- \in W$ in W, having the same edge length D_C. These square regions are selected in a manner such that there exists at least two robots $i, j \in \mathcal{R}$ with $b_i \in C^-$ and $b_j' \in C'^-$. Hence, each square contains at least one robot from one parent. Let M and M' denote the index set of robots in C^- and C'^- respectively as:

$$\begin{aligned} M &= \left\{ i \in \mathcal{R} | b_i^- \in C^- \right\} \\ M' &= \left\{ i \in \mathcal{R} | b_i'^- \in C'^- \right\} \end{aligned}$$

Following, two offsprings b^+ and b'^+ are generated by first duplicating each parent exactly, exchanging robots inside C^- and C'^- while the locations of the rest of the

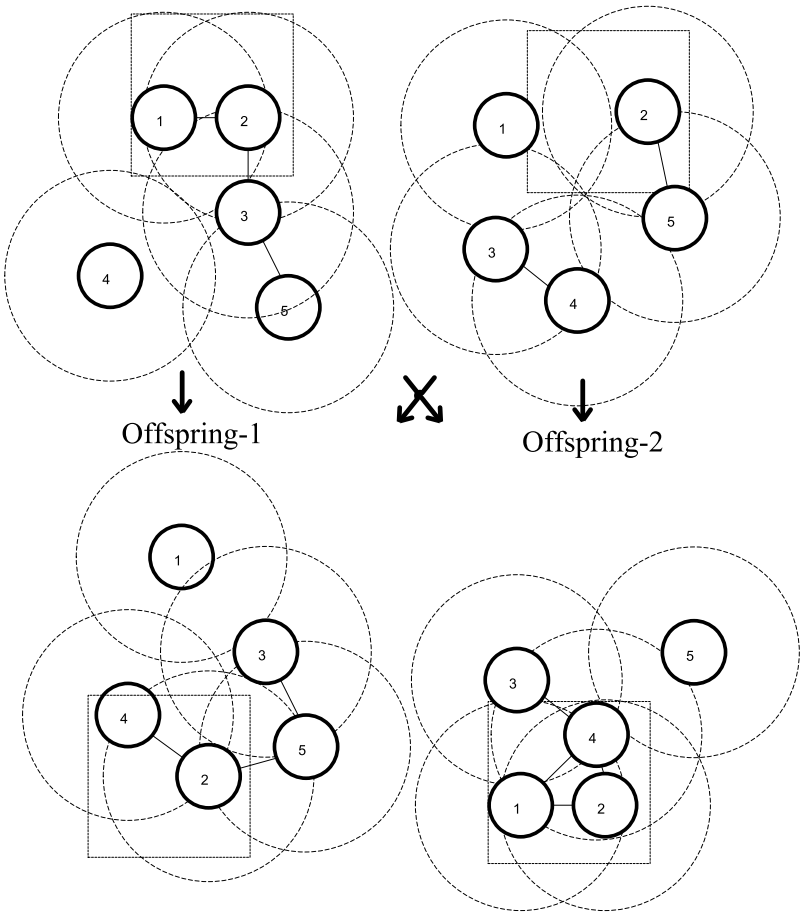


Figure F.2. Square crossover. In each graph, the robots are shown with dark circles and the communication ranges are shown with the dotted circles. If two robots's centers are within each others' communication range, a link is established as shown by the lines connecting their centers. The dotted circles indicate the squares selected. The two offsprings are generated via exchanging the robots.

robots are kept unchanged as much as possible. Furthermore, the centers of C^- and C'^- are moved to a new randomly selected location $c^+ \in W$ in the offsprings' workspace while ensuring that there is no collision. Of course, all the robots in the offsprings having indices identical to those in M and M' are removed. The offsprings b^+ and $b_i'^+$ are defined as follows:

$$b_i^+ = \begin{cases} b_i'^- - c^- + c^+ & \text{if } i \in M' \\ b_i^- & \text{if } i \notin M' \end{cases} \tag{F.10}$$

$$b_i'^{+} = \begin{cases} b_i^{-} - c^{-} + c^{+} & \text{if } i \in M \\ b_i'^{-} & \text{if } i \notin M \end{cases} \tag{F.11}$$

A sample square crossover is as shown in Figure F.2. Here, C^- contains robots 1 and 2 and hence $M = \{1, 2\}$. Similarly, C'^- contains robot 2 and hence $M' = \{2)$. Next, C^- and C'^- are moved to a new center. The two offsprings are generated via copying all the robots while exchanging those in M and M' respectively. The operation is completed after removing all the robots having identical indices with those in M and M'.

F.6.2. Mutation

The mutation operator is used to increase the variability of the population by perturbing each robots' position in a graph with a given probability. The algorithm uses two different kinds of mutation operators – robot and link mutations. At each iteration k, all the members of the population $S(k)$ are considered and only one type mutation is selected with probability p_R for robot mutation and p_L for link mutation.

If robot mutation is selected, robot mutation is applied on all the robots b_i in the given sample. There are two alternative operators depending on the restriction on the mutated location of each robot. The new position can be perturbed largely or slightly which correspond to the two types of operators. Each is selected with probabilities p_{R1} and p_{R2} respectively. For example, in Figure F.3a, the position of robot 3 is mutated with a large perturbation whereas in Figure F.3b, the same robot undergoes a slight perturbation.

If link mutation is selected, the mutations are applied on the links. There are three alternative operators and each is applied with probability p_{L1}, p_{L2} and p_{L3} respectively. First, a robot with one link only – known as leaf robot – is rotated through a random angle while perturbing the link length also slightly without breaking the link as shown in Figure F.4a for the link between robots 2 and 3. A second type of link mutation is where a randomly selected link is moved to a new location in W by keeping its length

and direction. An example is presented in in Figure F.4b where the link between robots 2 and 3 is translated. The third type of mutation is where a randomly selected robot that also does not have any links is forced to be connected to a randomly chosen nearby robot. An example is as seen in Figure F.4c where robot 1 is made to establish a link with robot 2.

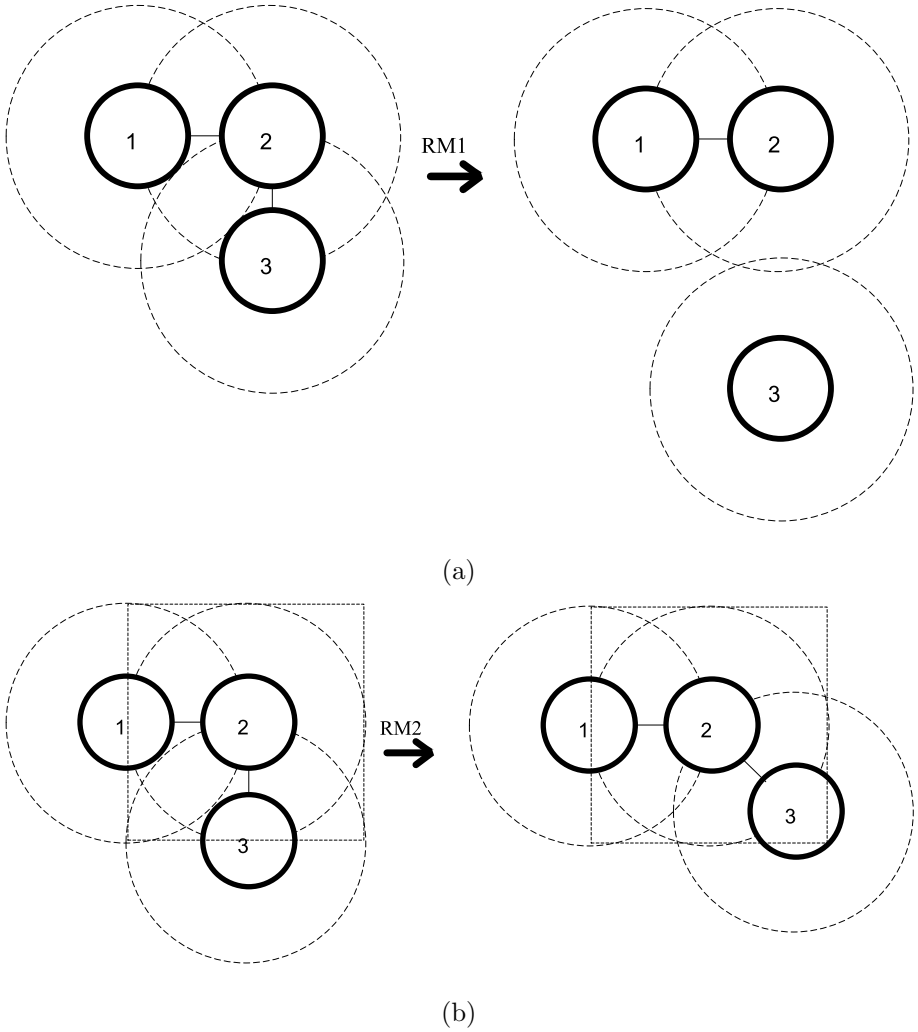


Figure F.3. Robot mutations. (a) Translation within the workspace; (b) Translation within a small area.

F.7. Simulations

In this section, we present simulation results from running the proposed algorithm for varying connectivity graphs. The values of all the parameters are set as presented in Table F.1. The type of the crossover operation is determined statistically as follows:

Table F.1. Parameters of the genetic algorithm.

Parameters	**Symbol**	**Value**
Workspace radius	ρ_0	50
Robot radius	ρ_i	0.5
Connectivity range	ρ_c	20
Population size	N_P	60
Max. number of generation	N_G	10000
Crossover probability	p_C	0.3
Crossover square size	D_C	15
RM probability	p_R	0.5
RM1 probability	p_{R1}	0.2
RM2 probability	p_{R2}	0.2
Mutation square size	D_M	5
LBM probability	p_L	0.5
LBM1 probability	p_{L1}	0.2
LBM2 probability	p_{L2}	0.3
LBM3 probability	p_{L3}	0.3
Constancy generation for elite	N_E	1000
Random new addition percentage	p_A	33
Best population percentage	p_B	10

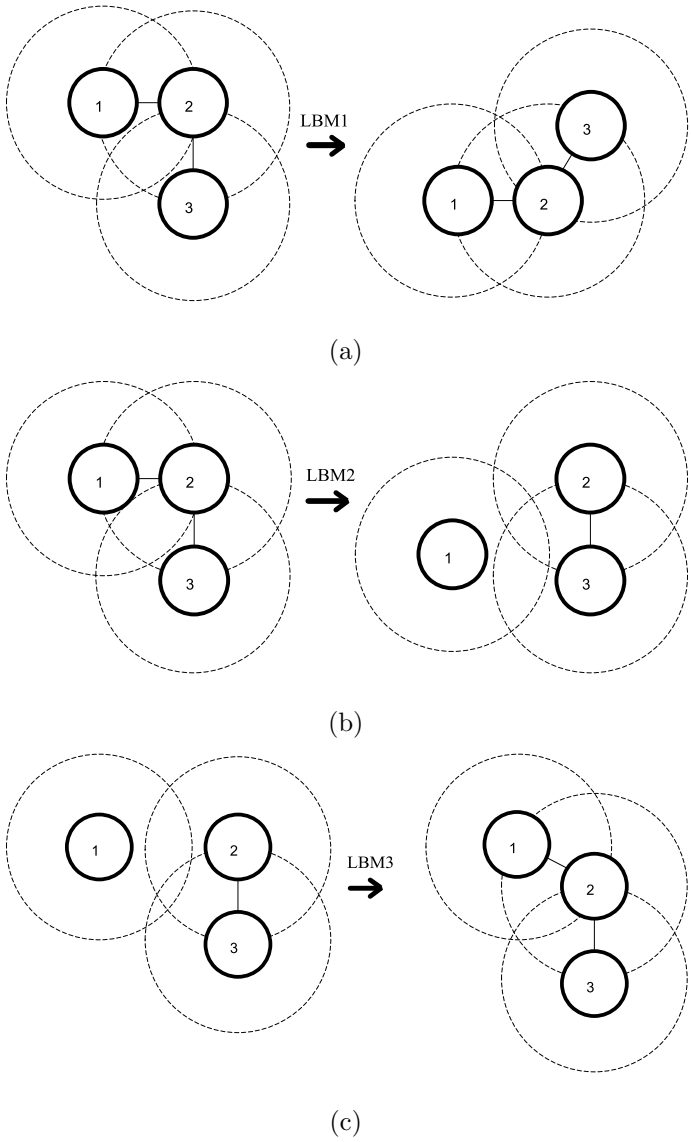


Figure F.4. Link-based mutations. (a) Rotation based; (b) Translation based; (c) Connectivity based.

We made 100 simulations with a high number of robots, identical parameter set and adjacency matrix and with different $S(0)$ using only one of crossover operators. The performance of each operator is assessed based on the percentage of realizations found in these simulations which turn out to be 62, 65 and 70 percent for the single-point, multi-point and square crossover operators respectively. Hence, in the remaining simulations, the only type of crossover operator used is the square crossover.

We start with $r = 8$ since an example of an unrealizable graph is the star $K_{1,7}$ with one central node connected to seven leaves as given by A_1. It is well known this

graph does not have a realization since using geometry, it can be seen that if each of seven unit disks touches a common unit disk, some two of the remaining seven disks must touch each other. A_2 is a realizable version of this graph. The algorithm does not generate a realization for A_1. For A_2, a sample graph is shown in Figure F.5.

$$A_1 = \begin{bmatrix} 0&1&1&1&1&1&1&1 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 0&1&1&1&1&1&1&1 \\ 1&0&1&0&0&0&0&1 \\ 1&1&0&1&0&0&0&0 \\ 1&0&1&0&1&0&0&0 \\ 1&0&0&1&0&1&0&0 \\ 1&0&0&0&1&0&1&0 \\ 1&0&0&0&0&1&0&1 \\ 1&1&0&0&0&0&1&0 \end{bmatrix}$$

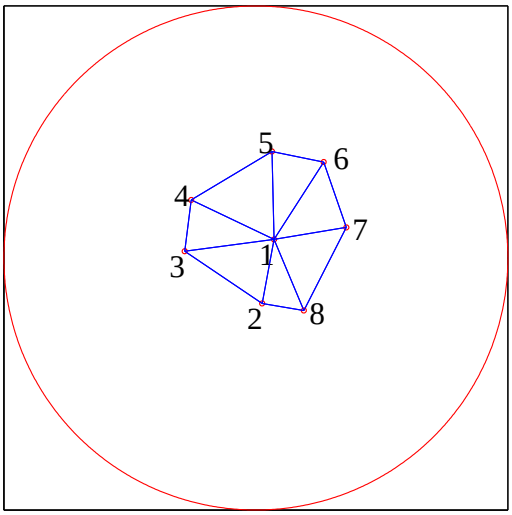


Figure F.5. A sample graph realization for 8 robots given A_2.

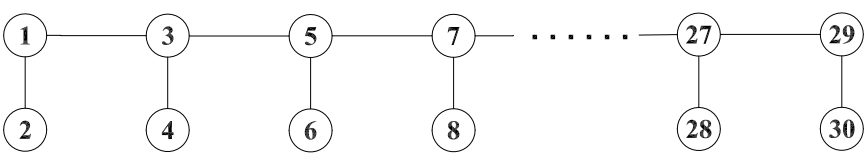


Figure F.6. A simple graph for 30 robots.

Next, we run the algorithm for $r = 30$ robots and seek realizations of a graph having degree 3 with a simple connectivity structure shown in Figure F.6. Due to size, the corresponding adjacency matrix is not provided. We run 100 simulations all with different initial population set $S(0)$. Six sample realizations generated by the algorithm are as shown in Figure F.7.

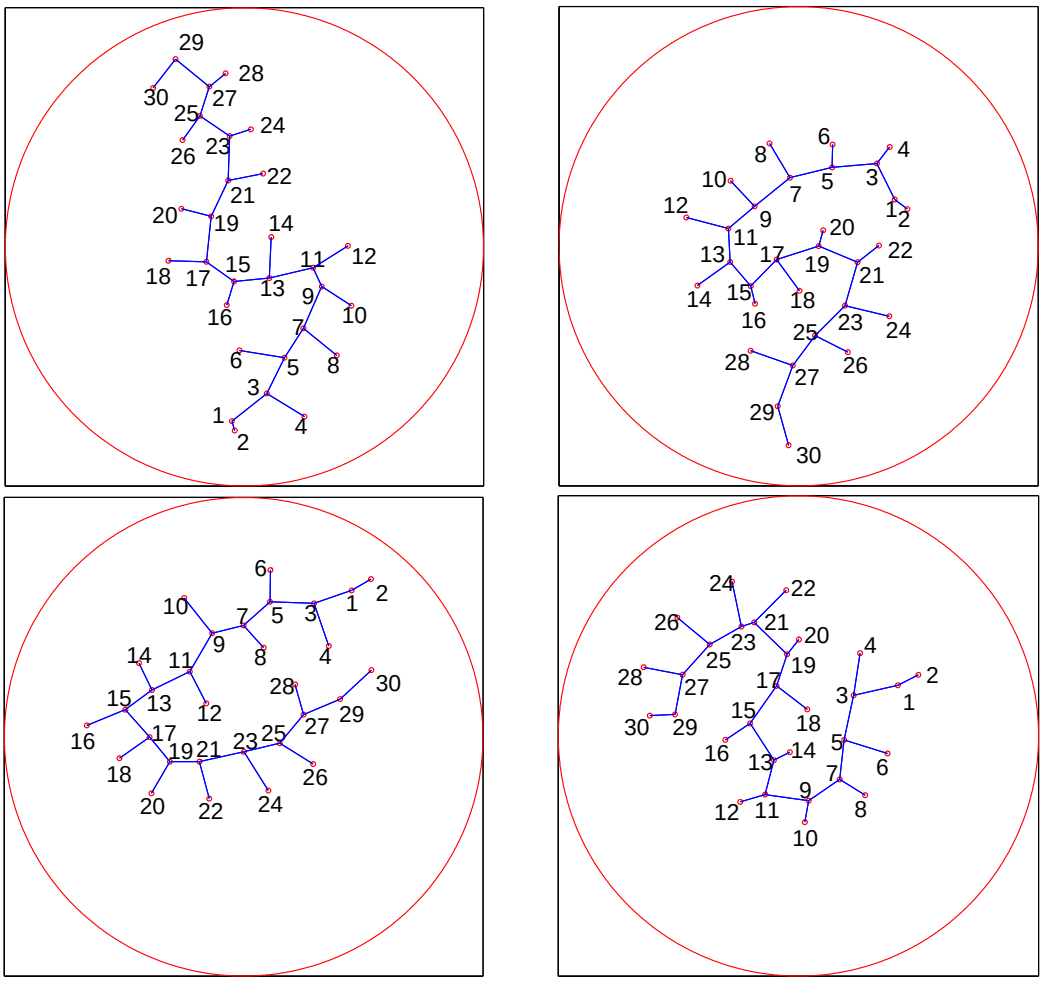


Figure F.7. Sample 30 robot realizations for a simple connectivity structure.

Finally, we run the algorithm again for $r = 30$ robots, but this time the realizations are of a graph having a more complicated connectivity as shown in Figure F.8. Again 100 simulations with different initial population set $S(0)$ are made. Six sample realizations generated by the algorithm are as shown in Figure F.9.

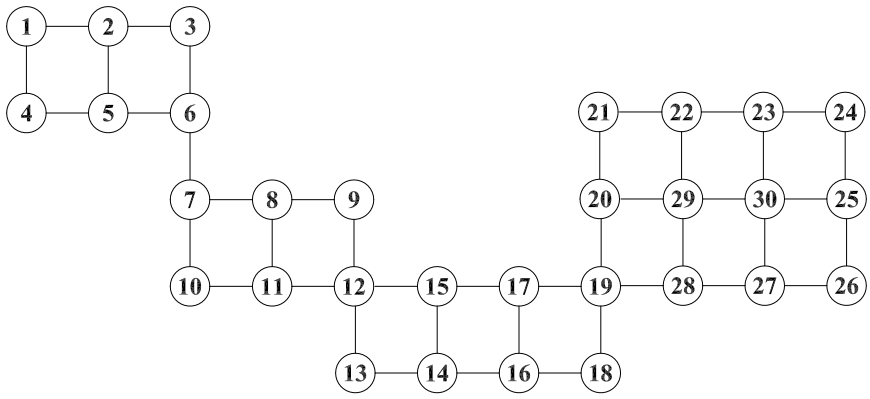


Figure F.8. A complicated connectivity graph for 30 robots.

F.8. Conclusion

This chapter considers the problem of planar multi-robot realizations of connectivity graphs. A realization is a set of robot locations in the planar workspace having a connectivity graph that is identical to an *a priori* given connectivity graph with the additional constraint it must be feasible. As the associated mathematical problem is known to be NP-hard, a stochastic approach based on genetic algorithms is proposed. Here, a population set is generated based on randomly generated feasible planar multi-robot positions. Each member in this set is then evaluated using a novel fitness function that measures the similarity of its connectivity graph with the given connectivity graph. New mutation operators that enable the evolution of generations are introduced. An extensive statistical study with different number of robots demonstrates that the proposed algorithm can be used to obtain realizations for fairly complicated connectivity graphs.

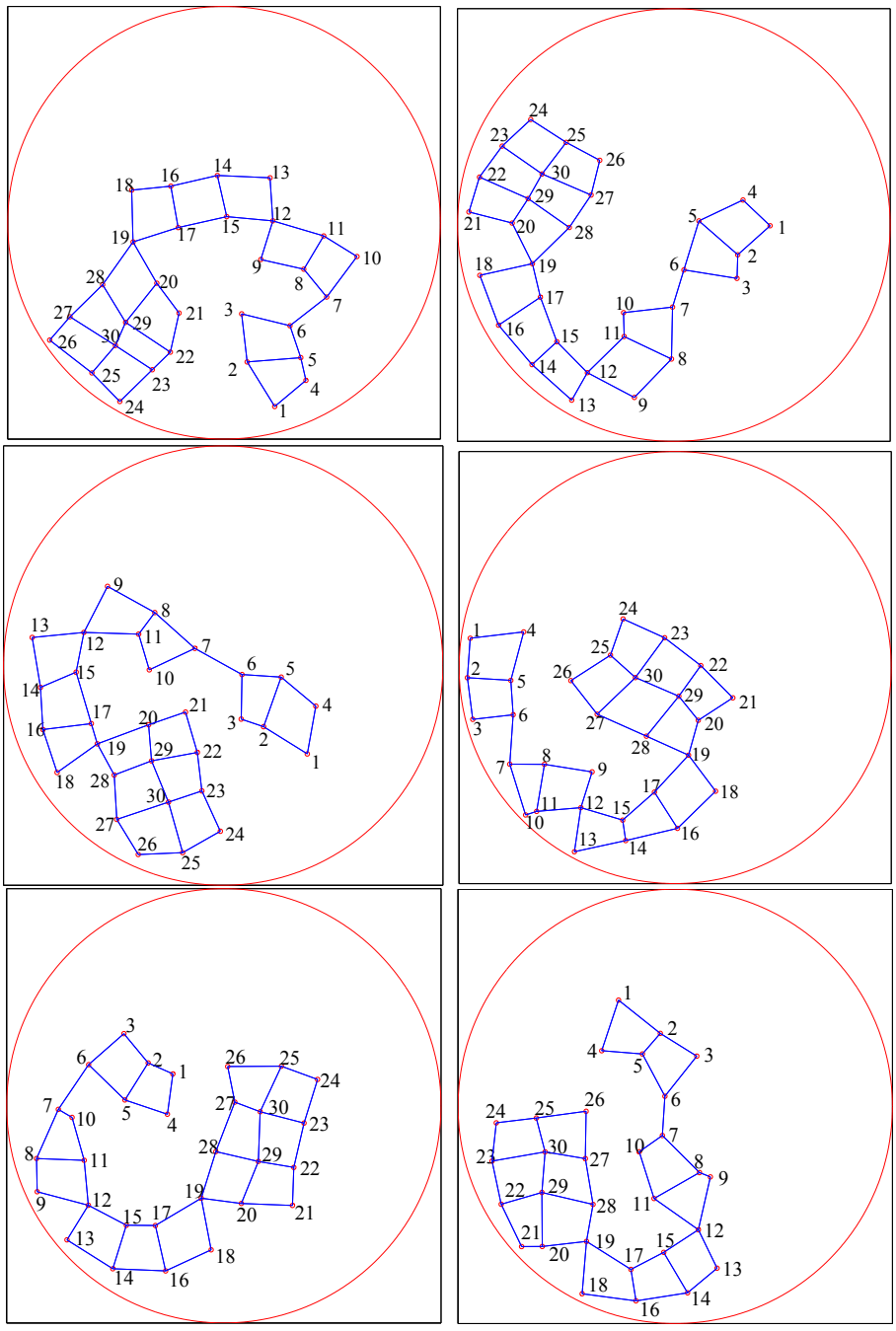


Figure F.9. Sample 30 robot realizations for the graph of Figure F.8.

APPENDIX G: CONNECTIVITY GRAPH BASED PLANAR MULTI-ROBOT DYNAMIC REALIZATION

This chapter deals with the problem of planar realizations of connectivity graphs in multirobot systems in a dynamic manner. The realization problem is defined as the generation of robot planar positions in which the corresponding connectivity graph is identical to a desired connectivity graph while being feasible. Feasibility means the robots must not be overlapping with each other. This problem can be also thought of as formation control problem and may arise in many applications such as exploration, search, patrolling and collective games (such as soccer) that require automatic positioning of multiple robots with a particular underlying connectivity graph constraint. For example, in multi-robot deployment and coordination tasks with limited communication, robot positions must satisfy the particular connectivity graphs [125].

G.1. Related Literature

The realization problem is related with the following research areas: (i) Disk graphs and graph drawing; and (ii) Formation control.

Connectivity graphs are known as unit disk graphs in graph theory – which are the intersection graphs of closed disks in the plane where each vertex corresponds to a circle and edge appears between two vertices when the corresponding circles intersect [128].A realization is therefore a mapping of the vertices to planar points which realize the graph [129]. The recognition problem of unit disk graphs is then posed as finding a realization for a given graph [129]. It has been shown that recognizing unit disk graphs is NP-hard. Nevertheless, the relevancy of this problem in many different application areas has motivated the development of approximate, force-directed [131,136] or stochastic approaches [132, 133]. An inapproximability result bounds on how well the planar coordinates can be derived from the connectivity information alone [130]. However, none of these algorithms enforce adherence to a given connectivity graph.

The realization problem has been addressed in [137] where both edges and nonedges are considered and a stochastic approach to the realization problem is proposed based on genetic algorithms.

Formation control has been extensively studied in the literature [138] where approaches are based mainly on optimization [139, 140] or potential fields [141–144]. The proposed approaches can be categorized depending on how a formation is defined and whether the formation has a leader or not. Formations are in general defined by either exact relative distances-orientations as is assumed in most work [143–148] or relative distances in inequality form [149]. While connectivity graphs [127] and maintaining connectivity [146] have been studied [149], the issue of attaining a realization for an arbitrary graph has been mostly overlooked.

G.2. General Approach

In this chapter, we present a novel approach to the planar graph realization problem based on a dynamical systems approach that is statistically proved to be working. First, the set of inequality constraints associated with the connectivity graph are transformed to equality constraints by defining proximity states in addition to robot configuration states. Following, an artificial potential function is constructed as to encode the equality constraints while ensuring feasibility which is then used to generate the control inputs that govern the robot and proximity state dynamics. The contribution of this work is that in contrast to our previous work in [137], it not only finds a realization of the given connectivity graph, but also simultaneously defines the control input that can be employed by the multirobot system in order to attain that realization from an arbitrary initial configuration. Hence, the approach may be applied in multirobot formation control applications. The outline of the Chapter is as follows: The rest of this Section presents robot states, connectivity and the problem statement. Section G.4 introduces the proximity state. The construction of the associated artificial potential function is explained in Section G.5. The robot and proximity dynamics are developed in Section G.6. A simulation study with different target graphs is presented in Section G.7. The Chapter concludes with a brief summary.

G.2.1. State Dependent Connectivity Graph

Suppose that the robots have limited sensing range ρ_s. A disk-neighborhood of robot i is a closed ball $B_{\rho_s}(b_i)$ of radius $\rho_s > \rho_i$ around $b_i \in \mathbb{R}^2$. Given ρ_s, any configuration $b \in \mathcal{F}$ induces a state-dependent mapping $g : \mathcal{F} \to \mathcal{G}$ as $g(b) = (\mathcal{R}, \mathcal{E}(b))$ - commonly known as connectivity graph. Here, $\mathcal{G} = \{g' | g' \subseteq K_r\}$ is the set of all possible (undirected) graphs on $\mathcal{R}$, K_r is the complete graph [24]. $\mathcal{E}(b) \subseteq Q$ is the set of edges defined as:

$$\mathcal{E}(b) = \{ij \in Q \mid \delta_{ij} \leq \rho_s\} \tag{G.1}$$

Each graph $g(b)$ is associated with a connectivity matrix $A(b) = [a_{ij}(b)]$ defined as follows:

$$a_{ij}(b) = \begin{cases} 1 & ij \in \mathcal{E}(b) \\ 0 & \text{otherwise} \end{cases} \tag{G.2}$$

Conversely, the relationship between $\mathcal{E}(b)$ and the connectivity matrix $A(b)$ is as follows:

$$\mathcal{E}(b) = \{ij \mid a_{ij}(b) = 1\} \tag{G.3}$$

G.3. Problem Statement

Consider a graph function $g : \mathcal{F} \to \mathcal{G}$. Suppose we are given target graph $g^* \in \mathcal{G}$. The set $g^{-1}(g^*) \cap \mathcal{F}$ represents the set of feasible robot configurations all having graph g^*. The goal is to find a realization $b^* \in g^{-1}(g^*) \cap \mathcal{F}$.

G.4. Proximity State

While a realization b^* is yet to be determined, its connectivity matrix $A(b^*) \equiv A^*$ is known - since g^* is given. In particular, let $A^* = \left\{a^*_{ij}\right\}$ denote the target connectivity

matrix with q components. Since b^* is not known, associated pairwise robot distances $\delta_{ij}^* = \| b_i^* - b_j^* \|$ is not known exactly. However, since A^* is given, they can be specified via a set of inequality constraints:

$$\delta_{ij}^* = \begin{cases} \leq \rho_s & \text{if} \quad a_{ij}^* = 1 \\ > \rho_s & \text{if} \quad a_{ij}^* = 0 \end{cases}$$

Further, if a realization b^* exists, then $\forall (i,j) \in Q$, $\exists\, \eta_{ij}^* \in \mathbb{R}^{\geq 0}$ such that δ_{ij} is exactly defined by

$$\delta_{ij}^{*\,2} = \begin{cases} \rho_s^2 - \eta_{ij}^{*\,2} & \text{if} \quad a_{ij}^* = 1 \\ \rho_s^2 + \eta_{ij}^{*\,2} & \text{if} \quad a_{ij}^* = 0 \end{cases}$$

Each η_{ij}^* specifies the actual proximity of the associate robot pair to the pairwise neighborhood boundary – namely $\delta_{ij} = \rho_s$ at the realization b^*.

Of course, η_{ij}^* are not known and need to be determined. Let η_{ij} denote the proximity state associated with the respective robot pair $(i,j) \in Q$. Feasibility constraints require that proximity states should be such that the resulting pairwise robot positioning remain within $\mathcal{F}$ while satisfying the inequality constraint. If $a_{ij}^* = 1$, then

$$0 \leq \eta_{ij}^2 < \rho_s^2 - \rho_{ij}^2$$

On other hand, if $a_{ij}^* = 0$, then

$$0 < \eta_{ij}^2 < (2\rho_0 - \rho_{ij})^2 - \rho_s^2$$

The admissible proximity space is defined as $\mathcal{E} \subset \mathbb{R}^q$ where constraints on all η_{ij} are all satisfied simultaneously. Define $\eta \in \mathcal{E}$ to be the proximity state vector $\eta = \left[\eta_{12} \ldots \eta_{(r-1)r}\right]^T$.

G.5. Artificial Potential Function

The artificial potential function $\widehat{\varphi} : \mathcal{F} \times \mathcal{E} \times \mathcal{G} \to [0, \infty]$ encodes both types of constraints – namely proximity constraints and feasibility constraints. It is defined as the ratio of two terms encoding these constraints as:

$$\widehat{\varphi}(b, \eta; g^*) = \frac{\gamma(b, \eta; g^*)^k}{\beta(b)} \tag{G.4}$$

The semicolon is used to indicate the parametric dependency of φ on g^* and $k \in \mathbb{Z}^+$ is the relative weighting parameter. It is constructed so that

$$\widehat{\varphi}(b^*, \eta^*) = 0$$

The function γ encodes the proximity constraint in a manner that depends on the robot configuration state, proximity state and the target graph. In particular, if $a^*_{ij} = 1$, then the distance between the robots should be within ρ_s. In contrast, if $a^*_{ij} = 0$, then the distance between the robots should be greater than ρ_s. This is mathematically expressed by γ_{ij}:

$$\gamma_{ij}(b, \eta; g^*) = a^\star_{ij} \left(\delta^2_{ij} - \rho^2_{c1}\right)^2 + (1 - a^\star_{ij}) \left(\delta^2_{ij} - \rho^2_{c2}\right)^2 \tag{G.5}$$

where $\rho^2_{c1} = \rho^2_c - \eta^2_{ij}$ and $\rho^2_{c2} = \rho^2_c + \eta^2_{ij}$. In this framework, each η_{ij} designates what the distance between each pair of robots (i, j) will be forced to be: Hence,

$$\gamma_{ij}(b, \eta; g^*) = \begin{cases} \left(\delta^2_{ij} - \rho^2_{c1}\right)^2 & \text{if} \quad a^\star_{ij} = 1 \\ \left(\delta^2_{ij} - \rho^2_{c2}\right)^2 & \text{if} \quad a^\star_{ij} = 0 \end{cases} \tag{G.6}$$

When $\gamma_{ij}(b, \eta; g^*) = 0$, then either $\delta^2_{ij} = \rho^2_{c1}$ or $\delta^2_{ij} = \rho^2_{c2}$ which means that the inequality

constraints are satisfied. As this needs to hold $\forall(i,j) \in Q$,

$$\gamma(b, \eta; g^*) = \sum_{(i,j)\in Q} \gamma_{ij}(b, \eta; g^*) \tag{G.7}$$

The denominator β encodes the distance from freespace boundary based on the robot state and the workspace restrictions. As the robots' positions cannot overlap, this can be encoded by a function $\beta_{ij} = \delta_{ij}^2 - \rho_{ij}^2$. Furthermore, as each robot needs to stay within the workspace, this restriction is encoded by $\beta_{0i} = \rho_{0i}^2 - \|b_i\|$. These two restrictions need to hold for all the robots and for all the robot pairs as encoded by the function β :

$$\beta(b) = \prod_{(i,j)\in Q^0} \beta_{ij} \tag{G.8}$$

where Q^0 denotes the index set of robot pairs including the workspace boundary as a *zeroth* disk, that is, $Q^0 \triangleq Q \cup \{(0,i)\,|\forall i \in \mathcal{R}\}$. Note that if $\exists(i,j) \in Q$, $\delta_{ij} = 0$, then $\beta(b) = 0$.

The function $\widehat{\varphi}$ is made admissible via composing it with $\sigma : [0,\infty) \to [0,1]$ defined as by $\sigma(x) = \frac{x}{x+1}$. In order to make the goal a non-degenerate critical point, further composition with a sharpening function $\sigma_d : [0,1] \to [0,1]$ defined as $\sigma_d(x) = x^{1/k}$ is used. Hence, the resulting artificial potential function $\varphi : \mathcal{F} \times \mathcal{E} \to [0,1]$ is defined as:

$$\varphi(b, \eta; g^*) = \sigma_d \circ \sigma \circ \widehat{\varphi}(b, \eta; g^*) \tag{G.9}$$

G.6. Robot & Proximity Dynamics

The robots start from an arbitrary initial condition $b(0)$ and $\eta(0)$. If $A(b) \neq A^*$, then they need to move both in $\mathcal{F}$ and $\mathcal{E}$ so that

$$\begin{aligned} \lim_{t\to\infty} b(t) &= b^* \\ \lim_{t\to\infty} \eta(t) &= \eta^* \end{aligned}$$

This is a dynamical formulation of the realization problem. While the robots are finding a realization satisfying the given constraints, they are guided by two dynamics. The first is related to robots' state b while the other is related to proximity state η. In this work, we let both be defined based on φ .

The robot dynamics govern how the robots will move in $\mathcal{F}$. It is defined by the negative gradient flow of φ on $\mathcal{F}$:

$$\dot{b} = -D_b\varphi(b, \eta; g^*) \tag{G.10}$$

with the initial conditions $b(0), \eta(0)$.

The proximity dynamics determine how the distance forced between the robot pairs will evolve. It is defined by the negative gradient flow of φ on $\mathcal{E}$

$$\dot{\eta} = -D_\eta\varphi(b, \eta; g^*) \tag{G.11}$$

again with the initial conditions $b(0), \eta(0)$.

Equation G.10 and Equation G.11 together define a set of coupled differential equations. Let $b^t(b(0), \eta(0))$ and $\eta^t(b(0), \eta(0))$ respectively denote the integral curve of $\dot{b}$ and $\dot{\eta}$ through the initial condition $b(0), \eta(0)$. The limiting sets $\lim_{t\to\infty} b^t(b(0), \eta(0))$

and $\lim_{t\to\infty} \eta^t(b(0), \eta(0))$ are defined by $D_b\varphi(b, \eta; g^*) = 0$.

$$\begin{aligned} -D_b\varphi(b, \eta^*; g^*) &= 0 \\ -D_\eta\varphi(b^*, \eta; g^*) &= 0 \end{aligned}$$

In differential games theoretic framework, this is known as the Nash equilibrium. Of course, each Nash equilibrium is not ensured of being a realization, unless $\varphi(b^*, \eta^*; g^*) = 0$.

G.7. Simulations

In this section, we present simulation results with the proposed approach for varying connectivity graphs. In our simulations, the workspace radius ρ_o is taken to be 10m and the sensing range ρ_s is set to 2m. All the robots are homogeneous – all having radii 25cm. A random initial configuration $b(0)$ for the robots is assumed.

We start with $r = 8$ since an example of an unrealizable graph is the star $K_{1,7}$ with one central node connected to seven leaves as given by A_1. It is well known this graph does not have a realization. Geometry dictates that if each of seven unit disks touches a common unit disk, two of the remaining seven disks must touch each other. A_2 is a realizable version of this graph. The robots exceed the maximum allowable time to generate a realization for A_1. For the realizable version A_2, a sample realization that is generated is shown in Figure G.1.

$$A_1 = \begin{bmatrix} 0\,1\,1\,1\,1\,1\,1\,1 \\ 1\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 0\,1\,1\,1\,1\,1\,1\,1 \\ 1\,0\,1\,0\,0\,0\,0\,1 \\ 1\,1\,0\,1\,0\,0\,0\,0 \\ 1\,0\,1\,0\,1\,0\,0\,0 \\ 1\,0\,0\,1\,0\,1\,0\,0 \\ 1\,0\,0\,0\,1\,0\,1\,0 \\ 1\,0\,0\,0\,0\,1\,0\,1 \\ 1\,1\,0\,0\,0\,0\,1\,0 \end{bmatrix}$$

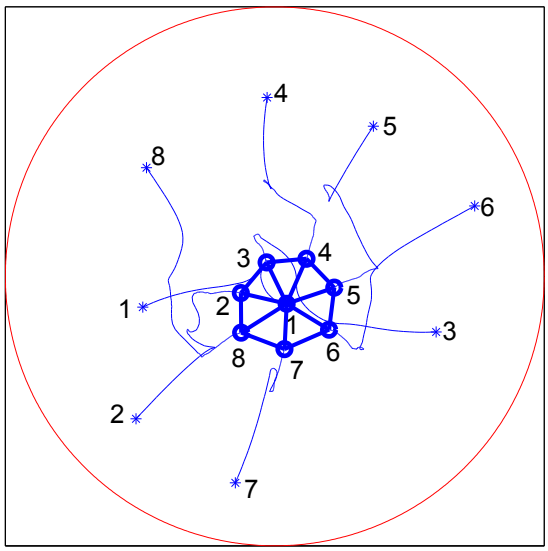


Figure G.1. A sample graph realization for 8 robots given A_2. The initial position of each robot i is shown with $*$. Open circle marks indicate the robots. Connections between robots are shown by thick straight lines while the trajectories are shown by thin lines.

Next, we consider simple unlabeled graphs of order equal to 7 since the cardinality of the associated partially order set (poset) is known to be equal to 1077, which means the total of 2^{21} possible different graphs can be partitioned into 1077 different cells where all graphs within the same cell are isomorphic to each other [150]. Motivated by a result that establishes the realizability of planar graphs[7] – namely graphs that can be drawn on a plane in such a way that there are no edge crossings [151] – embedded in the plane with maximum degree 4 [128], we limit our graphs to degree 4. This reduces our set to 509 graph types. The desired connectivity matrices A^* are generated for these graphs using an algorithm with two inputs: number of robots p and maximum degree m.

We then use our approach to generate realizations for all the generated 509 connectivity graphs A^*. The initial robot state $b(0)$ is randomly chosen as shown in Figure G.2. A sampled time evolution of the realization for the connectivity matrix $A^\star$

[7]Note that our graphs are not planar in general and hence this result is not applicable to our case in general.

in Equation G.12 is shown in Figure G.2.

$$A^\star = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{G.12}$$

The first figure shows the robots' initial positions. The next three graphs show the robots' positions at 25%, 50% and 75% of the total realization time. These figures show the progress of the motion along with the corresponding connectivity. The last figure shows the realization for the connectivity matrix $A^\star$. As shown in Table G.1, all graphs with degree 1 and 2 are successfully realized. However, there are some graphs with maximum degree 3 and 4 which cannot be realized since these graphs are most probably not realizable. In general, around 72% of all the graphs are successfully realized. As the problem is NP-hard, it is not possible to determine whether the rest is not realizable or the algorithm has terminated unsuccessfully.

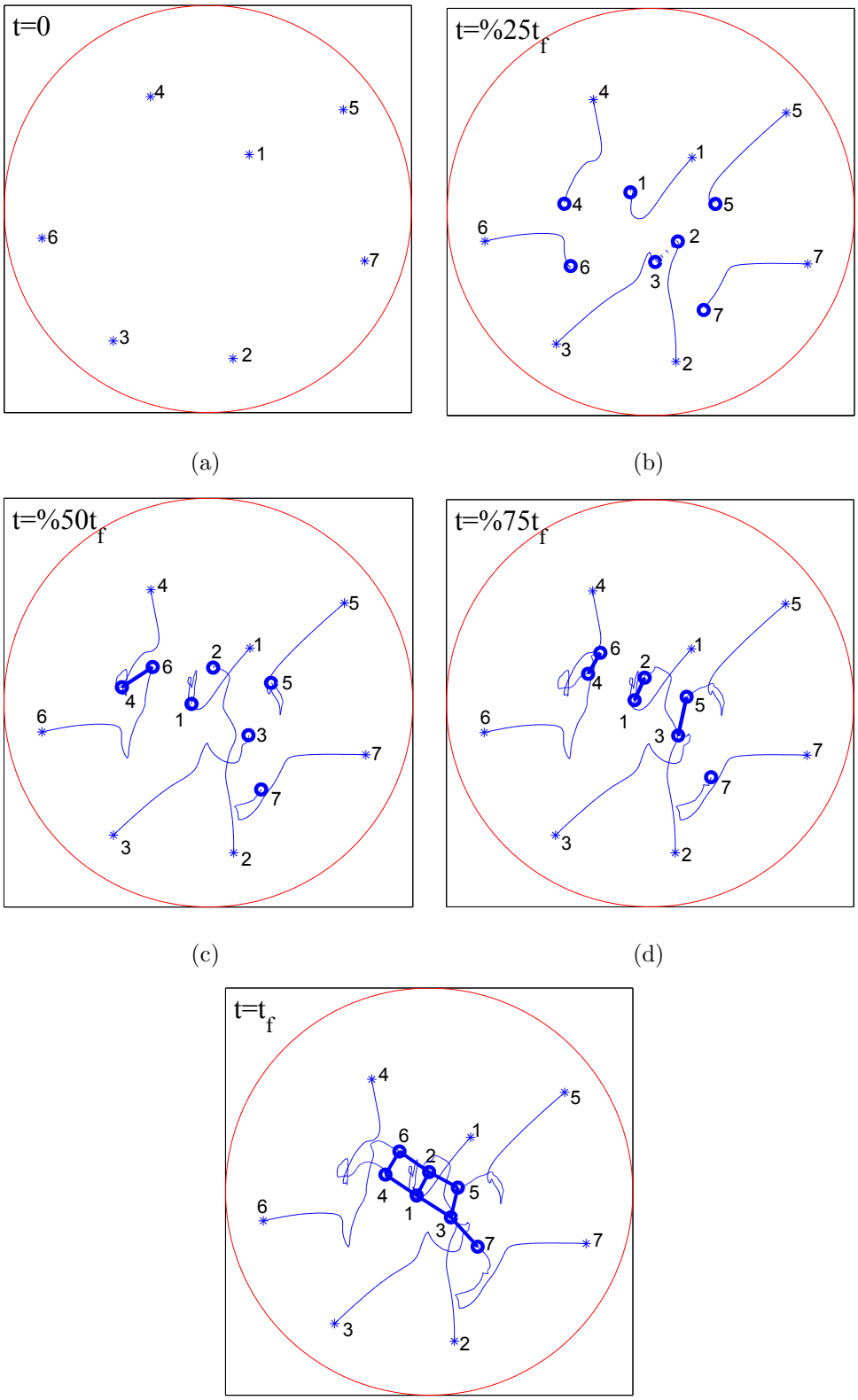
Table G.1. Simulation results for the realization.

Degree	Total graphs	Realized Graphs	Success %
1	3	3	100
2	25	25	100
3	121	102	84
4	360	240	66
Total	509	370	72

G.8. Conclusion

In this chapter, we consider the problem of planar multi-robot realizations of connectivity graphs. A realization is a set of robot locations in the planar workspace having a connectivity graph that is identical to an *a priori* given connectivity graph

with the additional constraint it must be feasible. The problem – known to be NP-hard - is associated with a set of inequality constraints that need to be satisfied simultaneously while ensuring feasibility. We propose a novel dynamical systems approach to this problem. First, inequality constraints are transformed into equality constraints via introducing proximity state. Next, an artificial potential function defined over the free robot configuration space and the admissible proximity space is used to define the robot and proximity state dynamics for attaining a realization. The advantage of this approach is that in contrast to previous work, it not only finds a realization, but also simultaneously defines the control input that can be employed by the multi-robot system to attain that realization. Hence, the approach may be used in multirobot formation control.



(a) (b) (c) (d) (e)

Figure G.2. 7 robot scenarios. (a) Random initial states $b_i(0)$; (b)-(e) The evolution of the realization with trajectories.

APPENDIX H: PUBLICATIONS RELATED TO THESIS

In the course of the PhD studies, the following papers have been published / submitted / presented:

Chapter 2

- H. Bayram and H. I. Bozma, "Multi-robot Navigation with Limited Communication - Deterministic vs Game-theoretic Networks", *IEEE/JRS International Conference on Intelligent Robots and Systems*, pp. 1825-1830, 2010.
- H. Bayram and H. I. Bozma, "Pairwise vs coalition game networks for multi-robot systems", *Proceedings of the 18 IFAC World Congress*, pp. 13570-13575, 2011.
- H. Bayram and H. I. Bozma, "Çoklu Robotlar için Algı Tümleşik Oyun-Kuramsal İletişim Ağı", *ASYU 2012 (Innovations and Aplications in Intelligent Systems Symposium)*, pp. 148-152, 2012.
- H. Karaoguz , O. Erkent, H. Bayram and I. Bozma I, "Tek Robottan Çoklu Robotlara Ortam Haritalama", *EMO Bilimsel Dergi*, vol. 4, no. 2, 2013.
- H. Bayram and H. I. Bozma, "Decentralized Network Topologies in Multirobot Systems", *Advanced Robotics*, 2013 (accepted).

Chapter 3

- H. Bayram and H. I. Bozma, "Multirobot Communication Network Topology via Centralized Pairwise Games", *IEEE International Conference on Robotics and Automation*, pp. 2506-2511, 2013.

Chapter 4

- H. Bayram and H. I. Bozma, "Assistance Networks for Dynamic and Cooperative Tasks in Multi-Robot Systems", *IEEE Transactions on Robotics*, 2013 (submitted).

Appendix E

- H. Karaoguz, H. Bayram, and H. I. Bozma, "Communication Integrated Control Architecture in Multirobot Systems", *ICRA Workshop on Towards Fully Decentralized Multi-Robot Systems: Hardware, Software and Integration*, 2013.

Appendix F

- H. Bayram and H. I. Bozma, "Planar multi-robot realizations of connectivity graphs using genetic algorithms", *IEEE/JRS International Conference on Intelligent Robots and Systems*, pp. 5163-5168, 2010.

REFERENCES

1. Yang, T., J. Ma, Z.-G. Hou, G. Peng and M. Tan, "A Multi-agent Architecture Based Cooperation and Intelligent Decision Making Method for Multirobot Systems", M. Ishikawa, K. Doya, H. Miyamoto and T. Yamakawa (Editors), *Neural Information Processing*, Vol. 4985 of *Lecture Notes in Computer Science*, pp. 376–385, Springer, 2008.

2. Rybski, P. E., S. A. Stoeter, M. Gini, D. F. Hougen and N. Papanikolopoulos, "Effects of Limited Bandwidth Communications Channels on the Control Of Multiple Robots", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 369–374, 2001.

3. Wei, L., "Stability Analysis Of Swarms with General Topology", *IEEE Transactions on Systems, Man and Cybernetics Part B*, Vol. 38, No. 4, pp. 1084–1097, 2008.

4. Horn, R. A. and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, New York, USA, 1991.

5. Arkin, R. C. and D. J, "Line-of-sight Constrained Exploration for Reactive Multiagent Robotic Teams", *7th International Workshop on Advanced Motion Control*, pp. 455–461, Maribor, Slovenia, 2002.

6. Rekleitis, I., V. Lee-Shue, A. P. New and H. Choset, "Limited Communication, Multi-robot Team Based Coverage", *IEEE International Conference on Robotics and Automation*, pp. 3462–3468, 2004.

7. Chaudhry, S., X. Xu and N. Amato, *Nearest Neighbor Search Methods*, Tech. rep., Department of Computer Science, Texas A&M University, 2008.

8. Dudek, G., M. Jenkin, E. Milios and D.Wilkes, "A Taxonomy for Multiagent

Robotics", *Autonomous Robots*, Vol. 3, No. 4, pp. 375–397, 1996.

9. Dudek, G., M. Jenkin and E. Milios, "A Taxonomy of Multirobot Systems", T. Balch and L. Parker (Editors), *Robot Teams: From Diversity to Polymorphism*, pp. 2–26, A.K. Peters, 2002.

10. Farinelli, A., L. Iocchi and D. Nardi, "Multirobot Systems: A Classification Focused on Coordination", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 34, No. 5, pp. 2015–2028, 2004.

11. Sanfeliu, A., N. Hagita and A. Saffiotti, "Network Robot Systems", *Robotics and Autonomous Systems*, Vol. 56, No. 10, pp. 793–797, 2008.

12. Goldsmith, A. J. and S. B. Wicker, "Design Challenges for Energy Constrained ad hoc Wireless Networks", *IEEE Wireless Communications*, Vol. 9, No. 4, pp. 8–27, 2002.

13. Kim, K.-H. and K. Shin, "On Accurate and Asymmetry-aware Measurement of Link Quality in Wireless Mesh Networks", *IEEE/ACM Transactions on Networking*, Vol. 17, No. 4, pp. 1172–1185, 2009.

14. Wellman, B. L., S. Dawson, J. Hoog and M. Anderson, "Using Rendezvous To Overcome Communication Limitations in Multirobot Exploration", *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2401–2406, 2011.

15. Klavins, E., "Communication Complexity of Multi-robot Systems", J.-D. Boissonnat, J. Burdick, K. Goldberg and S. Hutchinson (Editors), *Algorithmic Foundations of Robotics*, Vol. 5, pp. 275–292, 2004.

16. Diot, C., W. Dabbous and J. Crowcroft, "Multipoint Communication: A Survey of Protocols, Functions and Mechanisms", *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 3, pp. 277–290, 1997.

17. Das, S. M., Y. C. Hu, C. S. G. Lee and Y. Lu, "Supporting Many-to-one Communication in Mobile Multi-robot Ad Hoc Sensing Networks", *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 659–664, 2004.

18. Tang, L., K. Wang, Y. Huang and F. Gu, "Channel Characterization and Link Quality Assessment of IEEE 802.15.4 Compliant Radio for Factory Environments", *IEEE Transactions on Industrial Informatics*, Vol. 3, No. 2, pp. 99–110, 2007.

19. Yeo, I., K. J. Kim and A. Sunshin, "Performance Optimization Over Channel Model for A Low Rate Wireless Personal Area Network", *Proceedings of Third International Conference on Digital Information Management*, pp. 154–159, 2008.

20. Wu, X., B. S. Sharif, O. R. Hinton and C. C. Tsimenidis, "Solving Optimum TDMA Broadcast Scheduling in Mobile ad hoc Networks: A Competent Permutation Genetic Algorithm Approach", *IEE Proceedings - Communications*, Vol. 152, No. 6, pp. 780–788, 2005.

21. Chlamtac, I. and S. Kutten, "On Broadcasting in Radio Networks - Problem Analysis and Protocol Design", *IEEE Transactions on Communications*, Vol. 33, No. 12, pp. 1240–1246, 1985.

22. Tatikonda, S. and S. Mitter, "Control Under Communication Constraints", *IEEE Transactions on Automatic Control*, Vol. 49, No. 7, pp. 1056–1068, 2004.

23. Lygeros, J., K. H. Johansson, S. N. Simic, J. Zhang and S. S. Sastry, "Dynamical Properties of Hybrid Automata", *IEEE Transactions on Automatic Control*, Vol. 48, No. 1, pp. 2–17, 2003.

24. Mesbahi, M., "On State-dependent Dynamic Graphs and Their Controllability Properties", *IEEE Transactions on Automatic Control*, Vol. 50, No. 3, pp. 387–392, 2005.

25. Muhammad, A. and M. Egerstedt, "Connectivity Graphs as Models of Local Interactions", *Applied Mathematics and Computation*, Vol. 168, No. 1, pp. 243–269, 2005.

26. Martinez, S., F. Bullo, J. Cortes and E. Frazzoli, "On Synchronous Robotic Networks - Part I: Models, Tasks, and Complexity", *IEEE Transactions on Automatic Control*, Vol. 52, No. 12, pp. 2199–2213, 2007.

27. Piovesan, J. L., C. T. Abdallah and T. H. G, "Modeling Multi-agent Systems with Hybrid Interacting Dynamics", *Proceedings of American Control Conference*, pp. 3644–3649, 2009.

28. Bracka, P., S. Midonnet and G. Roussel, "Trajectory Based Communication in an Ad hoc Network of Robots", *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, Vol. 3, pp. 1–8, 2005.

29. Kravets, R., K. Calvert and K. Schwan, "Payoff-based Communication Adaptation Based on Network Service Availability", *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pp. 33–42, 1998.

30. Meshkati, F., H. V. Poor and S. C. Schwartz, "Energy-efficient Resource Allocation in Wireless Networks", *IEEE Signal Processing Magazine*, Vol. 24, No. 3, pp. 58–68, 2007.

31. Jiang, T. and J. S. Baras, "Fundamental Tradeoffs and Constrained Coalitional Games in Automatic Wireless Networks", *Proceedings of 5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc & Wireless Networks*, pp. 1–8, Limassol, Cyprus, 2007.

32. Arcaute, E., R. Johari and S. Mannor, "Network Formation: Bilateral Contracting and Myopic Dynamics", *IEEE Transactions on Automatic Control*, Vol. 54, No. 8, pp. 1765–1778, 2009.

33. Bayram, H. and H. I. Bozma, "Multi-Robot Navigation with Limited Communication - Deterministic vs Game-Theoretic Networks", *IEEE/RJS International Conference on Intelligent Robots and Systems*, pp. 1825–1830, Taipei, Taiwan, 2010.

34. Bayram, H. and H. I. Bozma, "Pairwise vs Coalition Game Networks for Multi-robot Systems", *Proceedings of the 18 IFAC World Congress*, pp. 13570–13575, Milan, Italy, 2011.

35. Ozgur, A., R. Johari, D. N. C. Tse and O. Leveque, "Information-theoretic Operating Regimes of Large Wireless Networks", *IEEE Transactions on Information Theory*, Vol. 56, No. 1, pp. 427–437, 2010.

36. Watts, A., "A Dynamic Model of Network Formation", *Games and Economic Behavior*, Vol. 34, pp. 331–341, 2001.

37. Jackson, M. O. and A. Wolinsky, "A Strategic Model of Social and Economic Networks", *Journal of Economic Theory*, Vol. 77, pp. 44–74, 1996.

38. Jackson, M. O. and A. Watts, "The Existence of Pairwise Stable Networks", *Seoul Journal of Economics*, Vol. 14, No. 3, pp. 299–321, 2001.

39. Jackson, M. O., *Social and Economic Networks*, Princeton University Press, Princeton, New Jersey, USA, 2010.

40. Jackson, M. O. and A. Watts, "The Evolution of Social and Economic Networks", *Journal of Economic Theory*, Vol. 106, pp. 265–295, 2002.

41. Bloch, B. and M. O. Jackson, "Definitions of Equilibrium in Network Formation Games", *International Journal of Game Theory*, Vol. 34, No. 3, pp. 305–318, 2006.

42. Calvo-Armengol, A. and R. Ilkılıç, "Pairwise-stability and Nash Equilibria in

Network Formation", *International Journal of Game Theory*, Vol. 38, pp. 51–79, 2009.

43. Bozma, H. and J. Duncan, "A Game-Theoretic Approach to Integration of Modules", *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 16, No. 11, pp. 1074–1086, 1994.

44. Chapman, A. C., A. Rogers, N. R. Jennings and D. S. Leslie, "A Unifying Framework for Iterative Approximate Best-response Algorithms for Distributed Constraint Optimization Problems", *Knowledge Engineering Review*, Vol. 26, pp. 411–444, 2011.

45. Basar, T. and G. J. Olsder, *Dynamic Noncooperative Game Theory*, SIAM, New York, USA, 1999.

46. Jackman, S., *Bayesian Analysis for the Social Sciences*, John Wiley and Sons, United Kingdom, 2009.

47. Chakrabarti, S. and R. P. Gilles, "Network Potentials", *Review of Economic Design*, Vol. 11, No. 1, pp. 13–52, 2007.

48. Dutta, B. and S. Mutuswami, "Stable Networks", *Journal of Economic Theory*, Vol. 76, pp. 322–344, 1997.

49. Alpcan, T., T. Basar, R. Srikant and E. Altman, "CDMA Uplink Power Control as a Noncooperative Game", *Wireless Networks*, Vol. 8, No. 6, pp. 659–670, 2002.

50. Kolar, V., S. Razak, P. Mahonen and N. Abu-Ghazaleh, "Measurement and Analysis Of Link Quality in Wireless Networks: An Application Perspective", *Proceedings of IEEE INFOCOM*, pp. 1–6, 2010.

51. Dhananjay, L., A. Manjeshwar, F. Herrmann, E. Uysal-Biyikoglu and A. Keshavarzian, "Measurement and Characterization of Link Quality Metrics in En-

ergy Constrained Wireless Sensor Networks", *Proceedings of Global Telecommunications Conference*, Vol. 1, pp. 446–452, 2003.

52. Gunturi, S. and F. Paganini, "Game Theoretic Approach to Power Control in Cellular CDMA", *Proceedings of IEEE Vehicular Technology Conference*, Vol. 4, pp. 2362–2366, 2003.

53. Han, Z., D. Niyato, W. Saad, T. Baṣar and A. Hjorungnes, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*, Cambridge University Press, Cambridge, UK, 2011.

54. Akyildiz, I. F. and M. C. Vuran, *Wireless Sensor Networks*, John Wiley and Sons, West Sussex, UK, 2010.

55. Leung, K. K., M. V. Clark, B. McNair, Z. Kostic, L. J. Cimini and J. H. Winters, "Outdoor IEEE 802.11 Cellular Networks: Radio and MAC Design and Their Performance", *IEEE Transactions on Vehicular Technology*, Vol. 56, No. 5, pp. 2673–2684, 2007.

56. Ramanathab, G. and V. S. Alagar, "Algorithmic Motion Planning in robotics: Coordinated Motion of Several Disks Amidst Polygonal Obstacles", *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 514–522, 1985.

57. LaValle, S. M. and S. Hutchinson, "Optimal Motion Planning for Multiple Robots Having Independent Goals", *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, pp. 912–925, 1998.

58. Simeon, T., S. Leroy and J. P. Lauumond, "Path Coordination For Multiple Mobile Robots: A Resolution-complete Algorithm", *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 1, pp. 42–49, 2002.

59. Kangasharju, J., S. Tarkoma and K. Raatikainen, "Comparing SOAP Perfor-

mance for Various Encodings, Protocols, and Connections", M. Conti, S. Giordano, E. Gregori and S. Olariu (Editors), *Personal Wireless Communications*, Vol. 2775 of *Lecture Notes in Computer Science*, pp. 397–406, Springer, 2003.

60. Bertsekas, D. and J. Tsitsiklis, "Some Aspects of Parallel and Distributed Iterative Algorithms - A Survey", *Automatica*, Vol. 27, No. 1, pp. 3–21, 1991.

61. Murray, J. and F. Stolzenburg, "Hybrid State Machines with Timed Synchronization for Multi-Robot System Specification", *Portuguese Conference on Artificial Intelligence*, pp. 236–241, 2005.

62. Schwager, M., N. Michael, V. Kumar and D. Rus, "Time Scales and Stability in Networked Multi-robot Systems", *IEEE International Conference on Robotics and Automation*, pp. 3855–3862, 2011.

63. Goyal, S. and S. Joshi, "Unequal Connections", *International Journal of Game Theory*, Vol. 34, No. 3, pp. 319–349, 2006.

64. Billand, P., C. Bravard and S. Sarangi, "A Note on Local Spillovers, Convexity, and The Strategic Substitutes Property in Networks", *Theory and Decision*, Vol. 75, No. 2, pp. 293–304, 2013.

65. Bayram, H. and H. I. Bozma, "Multirobot Communication Network Topology via Centralized Pairwise Games", *IEEE International Conference on Robotics and Automation*, pp. 2506–2511, 2013.

66. Bramoullé, Y. and R. Kranton, "Risk-sharing Networks", *Journal of Economic Behavior and Organization*, Vol. 64, pp. 275–294, 2007.

67. Lerman, K., C. Jones, A. Galstyan and M. J. Mataric, "Analysis of Dynamic Task Allocation in Multi-Robot Systems", *The International Journal of Robotics Research*, Vol. 25, No. 3, pp. 225–241, 2006.

68. Korsah, G. A., A. Stentz and M. B. Dias, "A Comprehensive Taxonomy for Multi-robot Task Allocation", *The International Journal of Robotics Research*, Vol. 32, No. 12, pp. 1495–1512, 2013.

69. Cao, Y., A. Fukunaga and A. Kahng, "Cooperative Mobile Robotics: Antecedents and Directions", *Autonomous Robots*, Vol. 4, No. 1, pp. 7–27, 1997.

70. Bererton, C. and P. Khosla, "An Analysis of Cooperative Repair Capabilities in a Team of Robots", *IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 476–482, 2002.

71. Parker, L., B. Kannan, F. Tang and M. Bailey, "Tightly-coupled Navigation Assistance in Heterogeneous Multi-robot Teams", *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 1016–1022, 2004.

72. Parker, C. A. C. and H. Zhang, "Collective Robotic Site Preparation", *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, Vol. 14, No. 1, pp. 5–19, 2006.

73. Gerkey, B. P. and M. J. Mataric, "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems", *The International Journal of Robotics Research*, Vol. 23, No. 9, pp. 939–954, 2004.

74. Zhang, Y. and L. E. Parker, "Considering Inter-task Resource Constraints in Task Allocation", *Autonomous Agents and Multi-Agent Systems*, Vol. 26, No. 3, pp. 389–419, 2013.

75. Ye, D., M. Zhang and D. Sutanto, "DGF: Decentralized Group Formation for Task Allocation in Complex Adaptive Systems", Q. Bai and N. Fukuta (Editors), *Advances in Practical Multi-Agent Systems*, Vol. 325 of *Studies in Computational Intelligence*, pp. 3–19, Springer, 2011.

76. Parker, L. E., "Decision Making as Optimization in Multi-Robot Teams", R. Ramanujam and S. Ramaswamy (Editors), *Lecture Notes in Computer Science*, Vol. 7154, pp. 35–49, Springer, 2012.

77. Zhang, K., E. G. Collins Jr. and D. Shi, "Centralized and Distributed Task Allocation in Multi-robot Teams via a Stochastic Clustering Auction", *ACM Transactions on Autonomous and Adaptive Systems*, Vol. 7, No. 2, pp. 21:1–21:22, 2012.

78. Wagner, T., J. Phelps and V. Guralnik, "Centralized VS. Decentralized Coordination: Two Application Case Studies", T. A. Wagner (Editor), *An Application Science for Multi-Agent Systems*, Vol. 10 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pp. 41–75, Springer, 2004.

79. Sandholm, T., K. Larson, M. Andersson, O. Shehory and F. Tohm, "Coalition Structure Generation with Worst Case Guarantees", *Artificial Intelligence*, Vol. 111, No. 12, pp. 209–238, 1999.

80. Sandholm, T., S. Suri, A. Gilpin and D. Levine, "Winner Determination in Combinatorial Auction Generalizations", *First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, pp. 69–76, 2002.

81. Lau, H. C. and L. Zhang, "Task Allocation via Multi-agent Coalition Formation: Taxonomy, Algorithms and Complexity", *15th IEEE International Conference on Tools with Artificial Intelligence*, pp. 346–350, 2003.

82. Vig, L. and J. A. Adams, "Coalition Formation: From Software Agents to Robots", *Journal of Intelligent and Robotic Systems*, Vol. 50, No. 1, pp. 85–118, 2007.

83. Dang, V. D. and N. R. Jennings, "Coalition Structure Generation in Task-Based Settings", *17th European Conference on Artificial Intelligence*, pp. 210–214, 2006.

84. Service, T. C. and J. A. Adams, "Coalition Formation for Task Allocation: Theory and Algorithms", *Autonomous Agents and Multi-Agent Systems*, Vol. 22, No. 2, pp. 225–248, 2011.

85. Chen, J. and D. Sun, "Resource Constrained Multirobot Task Allocation Based on Leader-follower Coalition Methodology", *The International Journal of Robotics Research*, Vol. 30, No. 12, pp. 1423–1434, 2011.

86. Cao, Y., A. S. Fukunaga and A. Kahng, "Cooperative Mobile Robotics: Antecedents and Directions", *Autonomous Robots*, Vol. 4, No. 1, pp. 7–27, 1997.

87. Yokoo, M., E. Durfee, T. Ishida and K. Kuwabara, "The Distributed Constraint Satisfaction Problem: Formalization and Algorithms", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 5, pp. 673–685, 1998.

88. Kephart, J. O., T. Hogg and B. A. Huberman, "Dynamics of Computational Ecosystems", *Physical Review A*, Vol. 40, pp. 404–421, 1989.

89. Weerdt, M., Y. Zhang and T. Klos, "Multiagent Task Allocation in Social Networks", *Autonomous Agents and Multi-Agent Systems*, Vol. 25, No. 1, pp. 46–86, 2012.

90. Jiang, Y., Y. Zhou and W. Wang, "Task Allocation for Undependable Multiagent Systems in Social Networks", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, No. 8, pp. 1671–1681, 2013.

91. Chapman, A. C., R. A. Micillo, R. Kota and N. R. Jennings, "Decentralised Dynamic Task Allocation: A Practical Game Theoretic Approach", *The 8th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 2, pp. 915–922, 2009.

92. Parker, L., "ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation", *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 2, pp.

220–240, 1998.

93. Gerkey, B. and M. Mataric, "Sold!: Auction Methods for Multirobot Coordination", *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, pp. 758–768, 2002.

94. McMillen, C., P. Rybski and M. Velosa, "Levels of Multi-Robot Coordination for Dynamic Environments", L. E. Parker, A. C. Schultz and F. E. Schneider (Editors), *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume III*, pp. 53–65, Springer, 2005.

95. Dias, M. B., R. Zlot, N. Kalra and A. Stentz, "Market-based Multirobot Coordination: A Survey and Analysis", *Proceedings of the IEEE*, Vol. 94, No. 7, pp. 1257–1270, 2006.

96. De Vries, S. and R. V. Vohra, "Combinatorial Auctions: A Survey", *INFORMS Journal on Computing*, Vol. 15, No. 3, pp. 284–309, 2003.

97. Koenig, S., C. Tovey, M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson and S. Jain, "The Power of Sequential Single-item Auctions for Agent Coordination", *Proceedings of the 21st National Conference on Artificial intelligence*, pp. 1625–1629, 2006.

98. Botelho, S. and R. Alami, "M+: A Scheme for Multi-robot Cooperation Through Negotiated Task Allocation and Achievement", *IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1234–1239, 1999.

99. Bernardine Dias, M. and A. Stentz, "Opportunistic Optimization for Market-based Multirobot Control", *International Conference on Intelligent Robots and Systems*, Vol. 3, pp. 2714–2720, 2002.

100. Brunet, L., H.-L. Choi and J. P. How, "Consensus-Based Auction Approaches for Decentralized Task Assignment", *AIAA Guidance, Navigation, and Control*

Conference, pp. 1–24, 2008.

101. Farinelli, A., L. Iocchi, D. Nardi and V. Ziparo, "Assignment of Dynamically Perceived Tasks by Token Passing in Multirobot Systems", *Proceedings of the IEEE*, Vol. 94, No. 7, pp. 1271–1288, 2006.

102. Bozma, H. I. and M. Kalalioglu, "Multirobot Coordination in Pick-and-place Tasks on a Moving Conveyor", *Robotics and Computer-Integrated Manufacturing*, Vol. 28, No. 4, pp. 530–538, 2012.

103. Karaoguz, H., H. Bayram and H. I. Bozma, "Communication Integrated Control Architecture in Multirobot Systems", *ICRA Workshop on "Towards Fully Decentralized Multi-Robot Systems: Hardware, Software and Integration"*, 2013.

104. Karagöz, C. S., H. I. Bozma and D. E. Koditschek, "Feedback-Based Event-Driven Parts Moving", *IEEE Transactions on Robotics*, Vol. 20, No. 6, pp. 1012–1018, 2004.

105. Fousse, L., G. Hanrot, V. Lefèvre, P. Pélissier and P. Zimmermann, "MPFR: A Multiple-precision Binary Floating-point Library with Correct Rounding", *ACM Transactions on Mathematical Software*, Vol. 33, No. 2, 2007.

106. Garey, M. R. and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, USA, 1979.

107. Shehory, O. and S. Kraus, "Methods For Task Allocation via Agent Coalition Formation", *Artificial Intelligence*, Vol. 101, pp. 165–200, 1998.

108. Hajdukova, J., "Coalition Formation Games: A Survey", *International Game Theory Review*, Vol. 08, No. 04, pp. 613–641, 2006.

109. Saad, W., Z. Han, M. Debbah, A. Hjorungnes and T. Basar, "Coalitional Game Theory For Communication Networks", *IEEE Signal Processing Magazine*, pp.

77–97, 2009.

110. Aumann, R. and J. Dreze, "Cooperative Games with Coalition Structures", *International Journal of Game Theory*, Vol. 3, No. 4, pp. 217–237, 1974.

111. Apt, K. R. and A. Witzel, "A Generic Approach to Coalition Formation", *International Game Theory Review*, Vol. 11, No. 03, pp. 347–367, 2009.

112. Bogomolnaia, A. and M. O. Jackson, "The Stability of Hedonic Coalition Structures", *Games and Economic Behavior*, Vol. 38, No. 2, pp. 201–230, 2002.

113. Saad, W., Z. Han, M. Debbah and A. Hjorungnes, "Coalitional Games for Distributed Collaborative Spectrum Sensing in Cognitive Radio Networks", *IEEE INFOCOM*, pp. 2114–2122, 2009.

114. Vig, L. and J. A. Adams, "Multi-Robot Coalition Formation", *IEEE Transactions on Robotics*, Vol. 22, No. 4, pp. 637–649, 2006.

115. Guerrero, J. and G. Oliver, "Multi-robot Coalition Formation in Real-time Scenarios", *Robotics and Autonomous Systems*, Vol. 60, No. 10, pp. 1295–1307, 2012.

116. Ducatelle, F., A. Förster, G. A. DiCaro and L. M. Gambardella, *Task Allocation in Robotic Swarms: New Methods and Comparisons*, Tech. Rep. IDSIA-01-09, IDSIA - Dalle Molle Institute for Artificial Intelligence, Lugano, Switzerland, 2009.

117. Labella, T., M. Dorigo and J.-L. Deneubourg, "Self-Organised Task Allocation in a Group of Robots", R. Alami, R. Chatila and H. Asama (Editors), *Distributed Autonomous Robotic Systems 6*, pp. 389–398, Springer, 2007.

118. Liu, W., A. F. T. Winfield, J. Sa, J. Chen and L. Dou, "Towards Energy Optimization: Emergent Task Allocation in a Swarm of Foraging Robots", *Adaptive Behavior*, Vol. 15, No. 3, pp. 289–305, 2007.

119. Apt, K. R. and T. Radzik, "Stable Partitions in Coalitional Games", *Arxiv Preprint cs/0605132*, pp. 1–8, 2006.

120. Ho, Y. and K. Chu, "Team Decision Theory and Information Structures in Optimal Control Problems-Part I", *IEEE Transactions on Automatic Control*, Vol. 17, No. 1, pp. 15–22, 1972.

121. Karagöz, C. S., H. I. Bozma and D. E. Koditschek, *Coordinated Navigation of Multiple Independent Disk-shaped Robots*, Tech. rep., The University of Michigan, Computer Science and Engineering Division, Department of Electrical Engineering and Computer Science, 2004.

122. Ali, S. and B. Mertsching, "Towards a Generic Control Architecture of Rescue Robot Systems", *IEEE International Workshop on Safety, Security and Rescue Robotics*, pp. 89–94, 2008.

123. Quigley, M., B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Ng, "ROS: An Open-source Robot Operating System", *ICRA Workshop on Open Source Software*, 2009.

124. Gunaratne, C., K. Christensen and B. Nordman, "Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP Connections, and Scaling of Link Speed", *International Journal of Network Management*, Vol. 15, No. 5, pp. 297–310, 2005.

125. Conner, M., M. U. Uyar, C. S. Sahin, E. Urrea, I. Hokelek, G. Bertoli and C. Pizzo, "Self-deployment of Mobile Agents in Manets for Military Applications", *Army Science Conference*, pp. 1–8, 2008.

126. Saber, R. and R. M. Murray, "Flocking with Obstacle Avoidance: Cooperation with Limited Communication in Mobile Networks", *Proceeding of 42nd IEEE Conference on Decision and Control*, pp. 2022–2028, 2003.

127. Muhammad, A. and M. Egerstedt, "Connectivity Graphs as Models of Local Interactions", *IEEE Conference on Decision and Control*, pp. 124–129, 2004.

128. Clark, B. N., C. J. Colbourn and D. S. Johnson, "Unit Disk Graphs", *Discrete Mathmatics*, Vol. 86, No. 1-3, pp. 165–177, 1990.

129. Breu, H. and D. Kirkpatrick, "Unit Disk Graph Recognition is NP-Hard", *Computational Geometry*, Vol. 9, pp. 3–25, 1998.

130. Kuhn, F., T. Moscibroda and R. Wattenhofer, "Unit Disk Graph Approximation", *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications(DIAL-M)*, pp. 17–23, 2004.

131. Kamada, T. and S. Kawai, "An Algorithm for Drawing General Undirected Graphs", *Information Processing Letters*, Vol. 31, pp. 7–15, 1989.

132. Eloranta, T. and E. Makinen, "TimGA: A Genetic Algorithm for Drawing Undirected Graphs", *Divulgaciones Matematicas*, Vol. 9, No. 2, pp. 155–171, 2001.

133. Zhang, Q.-G., H.-Y. Liu, W. Zhang and Y.-J. Guo, "Drawing Undirected Graphs with Genetic Algorithms", L. Wang, K. Chen and Y. Ong (Editors), *Advances in Natural Computation*, Vol. 3612 of *Lecture Notes in Computer Science*, pp. 28–36, Springer, 2005.

134. Sivanandam, S. and S. Deepa, *Introduction to Genetic Algorithms*, Springer, New York, USA, 2008.

135. Duda, R., P. Hart and D. Stork, *Pattern Classification*, John Wiley and Sons, New York, USA, 2001.

136. Lin, C., H. Yen and J. Chuang, "Drawing Graphs with Nonuniform Nodes Using Potential Fields", *Journal of Visual Languages & Computing*, Vol. 20, No. 6, pp. 385–402, 2009.

137. Bayram, H. and H. Bozma, "Planar Multi-robot Realizations of Connectivity Graphs Using Genetic Algorithms", *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5163–5168, 2010.

138. Murray, R., "Recent Research in Cooperative Control of Multi-Vehicle Systems", *Journal of Dynamic Systems, Measurement, and Control*, Vol. 129, No. 5, pp. 571–583, 2007.

139. Moore, B. J. and C. Canudas-de Wi, "Formation Control via Distributed Optimization of Alignment Error", *Proceedings of IEEE International Conference on Decision and Control*, pp. 3075–3080, 2009.

140. Dunbar, W. B. and R. M. Murray, "Distributed Receding Horizon Control for Multi-vehicle Formation Stabilization", *Automatica*, Vol. 42, No. 4, pp. 549–558, 2006.

141. Li, X. and Y. Xi, "Distributed Formation Algorithm for Multi-agent Systems with a Relaxed Connectivity Condition", *Proceedings of 17th IFAC World Congress*, pp. 5137–5142, 2008.

142. Krick, L., M. Broucke and B. Francis, "Stabilisation of Infinitesimally Rigid Formations of Multi-robot Networks", *International Journal of Control*, Vol. 82, No. 3, pp. 423–439, 2009.

143. Xue, D., J. Yao, G. Chen and Y.-L. Yu, "Formation Control of Networked Multi-agent Systems", *IET Control Theory & Applications*, Vol. 4, No. 10, pp. 2168–2176, 2010.

144. Gouvea, J., A. Pereira, L. Hsu and F. Lizarralde, "Adaptive Formation Control of Dynamic Nonholonomic Systems Using Potential Functions", *American Control Conference*, pp. 230–235, 2010.

145. Miklic, S., D.and Bogdan and R. Fierro, "Decentralized Grid-based Algorithms for

Formation Reconfiguration and Synchronization", *IEEE International Conference on Robotics and Automation*, pp. 4463–4468, 2010.

146. Dimarogonas, D. and K. Johansson, "Bounded Control of Network Connectivity in Multi-agent Systems", *IET Control Theory & Applications*, Vol. 4, No. 8, pp. 1330–1338, 2010.

147. Chen, J., D. Sun, J. Yang and A. Chen, "Leader-Follower Formation Control of Multiple Non-holonomic Mobile Robots Incorporating a Receding-Horizon Scheme", *The International Journal of Robotics Research*, Vol. 29, No. 6, pp. 727–747, 2010.

148. Sabattini, L., C. Secchi and C. Fantuzzi, "Arbitrarily Shaped Formations of Mobile Robots: Artificial Potential Fields and Coordinate Transformation", *Autonomous Robots*, Vol. 30, No. 4, pp. 385–397, 2011.

149. Zavlanos, M. and G. Pappas, "Potential Fields for Maintaining Connectivity of Mobile Network", *IEEE Transactions on Robotics*, Vol. 23, No. 4, pp. 812–816, 2007.

150. Adams, P., R. Eggleton and J. MacDougall, "Structure of Graph Posets of Orders 4 to 8", *Congressus Numerantium*, Vol. 166, pp. 63–81, 2004.

151. Harary, F., *Graph Theory*, Addison-Wesley, USA, 1994.