MULTI-ROBOT COORDINATION THROUGH MEDIATION OF INDEPENDENT DECISIONS

by

Soheil Keshmiri

M.Sc., Jamia Hamdard University, 2007

B.Sc., Pune University, 2004

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

in the
School of Engineering Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2012

APPROVAL

Name: Soheil Keshmiri

Degree: Doctor of Philosophy

Title of Dissertation: Multi-Robot Coordination through Mediation of Independent Decisions

Examining Committee: Dr. John Jones
Chair

Dr. Shahram Payandeh, Senior Supervisor
Professor, School of Engineering Science

Dr. Kamal Gupta, Supervisor
Professor, School of Engineering Science

Dr. Carlo Menon, Supervisor
Assistant Professor, School of Engineering Science

Dr. Mehrdad Moallem, Internal Examiner
Professor, School of Engineering Science

Dr. Hong Zhang, External Examiner
Professor, Computing Science, University of Alberta

Date Approved: August 31, 2012

Partial Copyright Licence

Abstract

This thesis contributes to the understanding of the intentional cooperation in multi-robot systems. We show that in a complex multi-robot system cooperation is achieved implicitly through the mediation of the independent decisions of the individual agents that are made autonomously. We assert that in order for a multi-robot system to accomplish a mission cooperatively, it is necessary for the individuals to proactively contribute to planning, to incremental refinement, as well as to adaptation at the group-level as the state of the mission progresses. We show how the decomposition of a mission delegated to a multi-robot system provides the agents with the ability to make decisions in a distributed fashion. While formulating decision mechanism, we show how the state of a robotic agent with respect to the delegated mission is viewed as two independent internal and external states. Furthermore, we demonstrate the requirement of *a priori* knowledge in decision process is prevented via incorporation of a simple sub-ranking module into the external state component of the decision mechanism of the robotic agents. While this independent involvement of the robotic agents in decision-making process preserves the autonomy of the individual agents, we show the mediation of these independent decisions does not only ensure proper execution of the plan but serves as a basis for the evolution of the intentional cooperation among the robotic agents.

To Sina, Shayan, and Sharmine

Acknowledgments

First and foremost, I would like to express my gratitude to my senior supervisor, Professor Shahram Payandeh, for his mentorship. He encouraged me to pursue the topic of the dissertation and allowed me to explore new frontiers through his inquiries and critical evaluation. Moreover, he provided me with a tremendous amount of insightful feedback on various versions of the dissertation. I would like to thank my committee members, Professor Kamal Gupta and Dr. Carlo Menon who significantly improved this work through their timely and valuable advice. I am grateful to my Internal Examiner Professor Mehrdad Moallem and my External Examiner Professor Hong Zhang for their thoughtful comments.

I would like to express my gratitude to the Interdisciplinary Research in the Mathematical and Computational Sciences (IRMACS) Centre for providing me with a vibrating environment in which to focus on my research and the technical capabilities to complete my defence.

I would like to thank my friend Dr. Reza Hosseini for the insightful conversations on various aspects of statistical analysis in the early stages of my research. I would like to express my gratitude to my friend Dr. Christopher Giles for his exceptional patience in various stages of editing the final dissertation and preparation of my defence.

I would like to express my deepest gratitude to my other friends who supported me throughout this journey. Mike Marin and Dr. Kyle Vincent desrve my praise. Ladan Hamadani was especially supportive of me when I needed it the most. Last but not least, I would like to express my gratitude to my mother, father, brothers, and sister who have supported me not just during my study, but also throughout my journey in life. I feel privileged to have you all in my life.

Contents

List of Tables

List of Figures

Chapter 1

Literature Review

Recent advancements in cooperative robotics have elevated the field within the robotics research community. Some of the advantages that can be achieved via deployment of such systems include improved system performance, distributed action at a distance and fault tolerance (Arkin, 1998, p. 360). Moreover, the real-life applications expand over a wide range of areas from gaming (e.g., Zhang et al., 2002), to exploration (e.g., Zlot et al., 2002), and surveillance (e.g., Amigoni et al., 2010) to search and rescue operations (e.g., Furukawa et al., 2006). In such systems, fulfillment of delegated missions is achieved through direct and/or indirect communication among agents. This results in a shift in the decision-making paradigm from a solitary activity into a process that inherently demands the engagement of all agents.

For a multi-agent system to reach the interconnectivity, interoperability and distributedness, required to preserve the autonomy of the agents to an acceptable degree, it is necessary for the agents to cooperate, reach agreement, or even compete with other systems/entities with conflicting/opposing interests. An immediate outcome of this perspective, is the necessity for agents to have the capacity to make proactive, real-time decisions, individually as well as collectively and as per group-level interest and performance.

Multi-robot systems are suitable for such domains when the overall task of the system is composed of several subgoals/tasks (e.g., foraging, rescue mission) or the task is monolithic in nature, but its fulfillment requires cooperative engagement of several agents (e.g., enclosing an intruder, box-pushing). Before investigating properties that are desirable and can be attributed to one such system, it is necessary to have a clear and descriptive definition of

what is referred to by the generic term *task*. This is due to the confusion that arises while using terms such as task, activity, and role. As Krieger and Billeter note:

> The words task, activity, and role may often be used one for the other as in "My task, activity, role is to sweep the yard". Still, *task* specifies "what has to be done", *activity* "what is being done", and *role* "the task assigned to a specific individual within a set of representatives given to a group of individuals. (Krieger and Billeter, 2000, p. 65)

Some researchers such as Oster and Wilson define a task as a set of behaviors that must be performed to achieve some purpose of the colony (Oster and Wilson, 1978, p. 326). Others follow a different approach and consider the term to be representative of an item of work that contributes potentially to fitness (Anderson and Franks, 2001, p. 534). Furthermore, they consider a subtask to be a part that makes a partial contribution to fitness once all other subtasks are completed (ibid., p. 534). In the context of multi-robot systems, Gereky and Mataric use the terms interchangeably since the underlying problem remains the same (Gerkey and Mataric, 2004b, p. 44). We follow same philosophy while referring to term *task*. Specifically and within the context of this dissertation, we consider a task to be a subgoal that is necessary for achieving the overall goal of the system, and that can be achieved independently of other subgoals (Gerkey and Mataric, 2004a, p. 939).

As a result, an inherent characteristic of a cooperative multi-robot system is the necessity for a robust decision-making and allocation strategy that distributes and redistributes available tasks among agents in an efficient way, given the system as well as delegated mission specifications.

Concepts such as efficiency are highly general, making it too complex to draw a definitive conclusion on what is an efficient performance while modeling a system. It may be asserted that efficiency is an optimization approach to the system performance. It is apparent in the context of this research that optimization is difficult to measure due to intractability of prioritization, in general and without any *a priori* assumptions, of interests of an agent or group of agents to the others. Therefore, some measures for estimating system performance are required.

Some strategies attempt to achieve optimality through deterministic modeling of system behavior, thereby bypassing the entire decision-making step. Others infer the expected behavior of the system at consecutive execution cycles through explicit reference to and direct

incorporation of contributions of agents to the overall performance of the system.

The remainder of the review chapter provides an overview of the research that focuses on multi-robot system decision-making and coordination. Despite the importance of the underlying architecture of such systems, the review is not intended to cover the various multi-robot system architectures. There exists a number of comprehensive studies and surveys that cover the topic (see Cao et al., 1997; Dudek et al., 2002). For instance, they provide a categorical taxonomy of available multi-robot systems that are classified along various directions such as team organization (e.g., centralized and/or distributed), communication topology (e.g., broadcast), as well as team composition (e.g., homogeneous versus heterogeneous).

1.1 Deterministic Modeling of System Behavior

Some approaches bypass the entire decision-making process through deterministic modeling of agent behaviors. Kube and Zhang (1996) formulate a box-pushing scenario as a sequence of subtasks with a separate controller designed for each step using finite state automata theory. There is no explicit communication among teammates. Robots obtain information about their environment through local perception and map their sensing to actuation in a reactive manner. Oh and Zelinsky (2000) and Silverman et al. (2002) program robots to attend a designated stationary docking station once they need recharging. This approaches to modeling of system behavior provides a limited solution to a small-scale team of robotic agents in a confined, well-organized field of operation. In addition, their limitations make them less appealing for medium and large teams of agents. Such short-comings vary from expending power for being recharged instead of performing their designated task(s) to determining the energy threshold for individuals as robots start spreading over the working environment. High level of traffic around the docking station is another drawback that is incurred while deploying such approaches. Munoz et al. (2002) address this issue by allowing robots to roam around the charging station for a short time once the high traffic is encountered.

While scenarios above are modeled around a single stationary recharging location, Zebrowski and Vaughan (2005) consider a mobile docking station. A mobile docking station achieves a higher scalability in terms of the spatial coverage and the size of robotic team. It also reduces the amount of energy expended by the agents for recharging. However, agents

are still instructed to attend one single designated location whose positioning information is changed temporally.

Litus et al. (2007) explores a distributed heuristic for an initially unknown single rendezvous location in an obstacle-free environment that minimizes the total traveling cost of the system. Apart from the single-rendezvous limit in this approach, complication of finding one such rendezvous location is increased as the worker population grows. Litus et al. (2009) utilize concept of Fermat-Torricelli point (see Boltyanski et al. (1999) for details) to overcome the single-rendezvous limitation of previous studies. It finds an optimal set of meeting places for the tanker to rendezvous with multiple worker robots for a given ordering of meetings.

In these approaches there is a one-to-one correspondence between robots and the meeting places. Agents are instructed to attend their corresponding meeting locations where positioning information is fully determined by the distribution of agents over the environment. In other words, agents are not permitted to make any modifications to their designated rendezvous locations, regardless of any further opportunities that become available to agents. As a result, no further decision is necessary and coordination is deterministically delivered through separation of agents tasks (i.e., disjoint meeting locations per agent and ordering of the meeting over time).

Pursuit and evasion scenarios are also addressed through the deterministic modeling of the game. Such approaches to the pursuit-evasion are classified into two main categories of differential and combinatorial techniques.

1.1.1 Differential Approach

The differential approach is based on non-cooperative differential games (see Basar and Olsder (1999) for details). It utilizes the solutions to differential equations of the motion of the players as control inputs to achieve the objective of the game. This approach further allows the physical constraints of robot (e.g., turning velocity and acceleration) to be incorporated into these equations. However, the complexity of the differential equations is proportional to the environmental complexity. This limits their scope to a locally or heuristically valid other than globally optimal solutions (Chung and Hollinger, 2011, p. 301).

Kim and Sugie (2007) introduce a target-enclosing strategy based on cyclic pursuit algorithms for a stationary target in an obstacle-free environment, modeled as 3D space. Guo et al. (2010) extend this approach to enclose a non-stationary target in a 2D space whose

velocity is piecewise constant but unknown to the pursuers. In these approaches the coordination of the mission is achieved by instructing every pursuer i to pursue *i+1 modulo n* pursuer, where n represents total number of pursuers. Undeger and Polat (2010) employ two coordination strategies to capture a prey. The blocking escape directions coordination calculates approaching directions of the predators to prey. It follows by the using alternative proposals to determine closest path of a predator to prey and in conjunction with the direction of the prey. As a result, the decision-making is simplified to finding which agent is the closest agent to the prey and coordination is instructing pursuers to follow the agent that is the closest to the prey.

1.1.2 Combinatorial Approach

The combinatorial techniques represent the environment geometrically using polygonal or graphical models. They directly employ these representations to address the pursuit games. Cops and robbers game originally introduced by Nowakowski and Winkler (1983) and Aigner and Fromme (1984) is a classic example of the graph-based approach to pursuit-evasion. In this setting, the game is accomplished if a cop moves onto the vertex occupied by a robber. The complexity of the algorithm for a single pursuer case is $O(n^4)$ and is exponential to the number of vertices and the pursuers in general. More specifically, the growth of the complexity of the algorithm is in order of $O(n^{2(k+1)})$ where n and k represent the number of pursuers and vertices of the graph, respectively. Isler and Karnad (2008) show that the duration of the game is bounded by the number of vertices in case of a complete graph. However, the representation of the environment in the form of a complete graph K_n is an oversimplification of the problem. This is in particular true if the behavior of the intruder is deterministic. More specifically, in a K_n graph one single move of the pursuer is sufficient to capture the intruder.

Jankovic (1978) shows that in a polygonal environment three pursuers are sufficient to capture an intruder if the initial location of the intruder is within the convex hull of the locations of the pursuers. Kopparty and Ravishankar (2005) generalize this result and prove that the number of pursuers is proportional to the dimensions of the environment. They show that in a $\mathbb{R}^d$, $d \geq 2$, polygonal environment, $d + 1$ pursuers achieve the same result. Isler et al. (2005) demonstrate that a pursuer equipped with a randomized strategy can locate an intruder in any simply-connected polygon. Bopardikar et al. (2007) study the pursuit-evasion under limited sensing condition where the sensing capability of the players

are limited within a threshold range. However, the intruder exhibits a reactive behavior and changes its location only if it senses the presence of a pursuer. Thunberg and Orgen (2010) use mixed linear integer programming approach to address a visibility-based pursuit scenario in a polygonal environment.

One of the limitation of the combinatorial approaches is due to the representation of the paths of the players as edges of the graph. Chung and Hollinger note that:

> a large portion of fundamental work in pursuit-evasion examined the problem of edge search, where the evader resides on the edges of a graph. Edge search does not apply directly to many robotics problems. The possible paths of an evader in many indoor and outdoor environments often cannot be accurately represented as the edges in a graph. In some cases, it is possible to construct a dual graph by replacing the nodes with edges, but these translations do not necessarily yield the same results as the original problem. (Chung and Hollinger, 2011, p. 310)

In addition, the polygonal and the graphical representations suffer from the free movement of the players among available nodes. In other words, players are allowed to move between nodes without following edges (ibid., p. 310). The comprehensive surveys of combinatorial approaches are found in (Alspach, 2004; Fomin and Thilikos, 2008).

1.2 Decision-Making and Agents Utility

Since the coordination of multi-robot system is essentially the comparative analysis of selective conclusions on available alternatives, within multi-robot system research community the *utility* or the *fitness function* (a.k.a *payoff function*) plays an influential and prevailing role. Individual estimates on costs and rewards associated with execution of the delegated tasks provide a measure based on which conclusion on optimal performance of system is determined. A wide variety of approaches ranging from planner-based techniques (e.g., Botelho and Alami, 1999) to simple reactive and sensor-based methodologies (e.g., Gerkey and Mataric, 2002) are adapted to formulating the utility functions.

Beil and Vaughan (2009) present a moving charging station capable of determining its best location within agents working environment based on the robotic agents density and work flow. The approach uses the proximity of an agent to the recharging dock D, its navigability score N, and agent velocity v to formulate votes of the agents in form of the tuple

$\{D, N, v\}$. In this context, navigability refers to the agent progress towards traversing its designated task or dock other than avoiding obstacles. The vector sum of the values of the tuples indicates the fitness of locality of the docking station within the field and with respect to the robotic team distribution as well as workload. However, to find one such location, the charging station requires an exhaustive search over the robots work space. Moreover, the searching process is repeated once the density of agents is changed from one part of the environment to another.

In matrix games (e.g., Basar and Olsder, 1982) utility functions are represented as payoff matrices, one per agent, in which the joint actions of agents are calculated. Joint actions correspond to particular entries in the payoff matrices and agents play the same matrix games repeatedly. The gain and losses of the system through these actions is calculated and if the outcome does not violate some predefined criteria the agents are accordingly assigned to their corresponding actions.

In the works of Levy and Rosenschein (1992) and Harmati and Skrzypczyk (2009) utility functions are used in a game-theoretic framework to incorporate the global goal of the system into agents that are obtained locally. The utilities are generally used to achieve a predetermined stable state such as the equilibrium of the system.[1] For instance, in a multi-robot target pursuit, the controller picks the combination of actions in which total distance traveled by the agents at the system-level and at every iteration is minimized. As a result, the equilibrium or stable-state is travel distance that is within a threshold condition. An important limitation of the game-theoretic approach to utility functions is the lack of guarantee of optimization of the proposed solution on a given decision cycle. The sole concern of the controller is to avoid the loss by the agent (e.g., to undergo more travel distance) at a predetermined value, or gain above the given threshold in the maintenance of stability.

Bayesian formalism is widely used for modeling the utility functions (e.g., Chalkiadakis, 2003). It utilizes the measurement (e.g., distance, sensor reading) and the control data (e.g., actuator, end-effector) to calculate the belief distribution of an agent in conjunction with a given task. It is a recursive algorithm where the belief at time t is calculated based on its value at time $t-1$ (Thrun et al., 2006, p. 27). In this framework, decision-makers use priors to reason the manner in which their actions influence the behaviors of other agents. These priors vary from the probability distribution function of resource densities over the

[1] A robotic system is considered to meet the equilibrium condition if there is a dynamic working balance among its interdependent parts (see Shoham and Brown (2009) for further details).

field of operation to the preassigned ranking of different actions available to agents. As a result, some prior densities over possible dynamics (e.g., the probability of agent r_j to take action a_i if the current decision-maker r_1 chooses a_1) and reward distributions have to be known by a decision-maker in a priori.

Seminal works by Koopman (1956a,b) outline analytical principles for applied probability and optimization techniques for maritime warfare strategies. Assaf and Zamir (1985) utilize the distributions of the locations of multiple objects to construct the priors distributions in Bayesian framework. Washburn (1983) presents an iterative algorithm based on Morkov decision process (MDP) to pursue a target in a discretized search environment.

The basic assumption in probabilistic framework is the ability of an agent to observe the actions taken by all agents, the resulting outcomes, and the rewards received by other agents. In this framework an agent incorporates information from the actions of other agents. In formulating their strategy, it is assumed that agents can keep track the history of their as well as the previous actions of teammates to make future decisions. Furthermore, the probabilistic approaches are applicable only for special distributions and rely on the absence of false positive detection errors (Chung and Hollinger, 2011, p. 308). Additionally, they require an analytical specification of *a priori* information (e.g., probability distribution of the location of the target within the field). Furthermore, these approaches require a measure of the density of the search (e.g., search time, resources expended) and the probability of detection given this density.

It has been noted by Brown and Shoham (2008, p. 4) that there is subjectivity in validating the desirability of an outcome in a solution space to the others. More specifically, the application of any positive affine transformation to the utility function of an agent results in another valid utility. However, such subjectivity comes with the trade-off of the flexibility of the utility formulation. It provides multi-robot systems with the capability of incorporating wide ranges of criteria and arbitrary computational steps to calculate the agents as well as the overall system gains and losses once in a mission. Despite such subjectivity and relativity of the computation, utilities provide the necessary means to analyze the optimality of system performance:

> Regardless of the method used for calculation, the robots' utility will be inexact due to sensor noise, general uncertainty, and environmental change. These unavoidable characteristics of the multi-robot domain will necessarily limit the efficiency with which coordination can be achieved...When we discuss "optimal"...we mean...given the union of all information available...it is impossible to construct a solution with higher overall utility. (Gerkey and Mataric, 2004a, p. 941)

1.3 Coordination

Estimating costs and rewards at the individual-level is the first step for achieving cooperative behavior among teammates. In order to devise agents with the capability of performing delegated tasks reliably while expending least resources possible and accumulating maximum achievable rewards, the coordination of individual decisions is paramount. An exhaustive body of literature in research is dedicated to the multi-robot coordination and task-allocation problem. It ranges from the greedy algorithm (Parker, 1998) and optimization techniques (Atay and Bayazit, 2006) to auction/market-based approaches (Bertsekas, 1990) and biologically-inspired methodologies (Walker and Wilson, 2011). Dahl et al. (2009) show that multi-robot coordination and task allocation in systems with significant performance effects from group dynamics falls in NP-complete domain. A comprehensive survey of the topic is found in (Gerkey and Mataric, 2004a; Campbell and Wu, 2011).

Market-based approaches are gaining popularity among methodologies that are adapted for coordinating multi-robot systems. Auctions are the most common mechanisms used in such formalism (see Wolfstetter (1996) for details on auction mechanism). In an auction, a set of items (e.g., tasks to be performed by individual agents) is offered by an auctioneer in an announcement phase. The participants make offers for these items by submitting bids to the auctioneer. Once all bids are received or a specific deadline is passed, the auction is cleared in the winner determination phase by the auctioneer who decides which items to award and to specific agents.

Market-based approaches fall into several sub-categories, including combinatorial auctions (e.g., Berhault et al., 2003; Nair et al., 2002), central single-task iterated auctions (e.g., Lagoudakis et al., 2005; Tovey et al., 2005), central instantaneous assignment (see Gerkey and Mataric, 2002, and Kose et al., 2004), peer-to-peer trading (e.g., Rabideau et al., 1999; Zlot et al., 2002), or combination of central multi-task auctions and peer-to-peer trading (

e.g., Dias et al., 2003). Most existing market-based approaches in the allocation strategies taxonomy are classified under single-robot/single-task (SR-ST) category. Whereas some approaches permit instantaneous assignments only (e.g., Gerkey and Mataric,2002; Kose et al., 2004), other approaches allow time-extended allocation in the form of sequencing (e.g., Lagoudakis et al., 2005; Berhault et al., 2003; Rabideau et al., 1999) or task scheduling (e.g., Lemaire et al., 2004; Schenider et al., 2005).

In a multi-robot coordination context, instantaneous allocation is that available information concerning the robots, the tasks, and the environment allows the system to perform the assignment of tasks only once with no planning for future allocation (Gerkey and Mataric, 2004a, p. 943). In contrast, time-extended allocation (e.g., Dias and Stentz (2001,2002); Nanjanath and Gini, 2010) refers to the strategies in which it is assumed that a model of how tasks are expected to arrive over time is available.

Dias et al. (2006) show that in a limited number of rounds, a combination of single- and multi-task trades outperforms all other combinations of single-task, swap, and multi-party contracts (e.g., Andersson and Sandholm, 2000). Furthermore, they claim that increasing the maximum number of tasks awarded per multi-task auction results in a poorer solution quality.

It has been noted that existing market mechanisms are not fully capable of re-planning tasks distribution, changing decomposition of tasks, and re-planning coordination during execution. Additionally, scalability is another issue that limits the applicability of market-based approaches for complex scenarios:

> Scalability in the market-based approaches may be limited by the computation and communication needs that arise from increasing auction frequency, bid complexity and planning demand. (Talay et al., 2011, p. 330)

For instance, it is apparent that instantaneous allocation is not a reliable choice for tasks whose positioning information change in time/space. Examples of such missions vary from rescue missions to spill perimeter detection and surveillance in hazardous environments. In addition, such approaches are unable to fully account for situations in which agents have different probabilities of success (i.e., successful completion of delegated tasks over time/space). For example, in a pursuit mission an agent that has higher chance of capturing the intruder in one decision cycle, might be a better choice of blocking a escape direction or exit way in another. This is due to a series of events that have been taken place between consecutive

decision cycles. Such events vary from change in the heading direction or behavior of the intruder to teammates that were forced to take a detour to avoid collision and appearance of an unforeseen obstacle along the path of the agent. Sandholm (2002), Dias and Stentz (2002), and Nanjanath and Gini (2010) utilize the time-extended methodology to address the latter issue.

Parker (1998) devises an algorithm that performs iterated allocation by assigning tasks that are learned through experience. In this approach, robots coordinate their respective actions explicitly through deliberate communications and negotiations. Botelho and Alami (2009) utilize contract net protocol to formalize task-allocation strategy that requires predefinition of robots capabilities and costs. This presumption limits the scope as well as applicability of the approach. Werger and Mataric (2001) demonstrate a methodology to determine locally decided eligibilities for task-allocation based on a fully connected inter-robot communication network. However, a fully connected robotic mesh increases the computational complexity as well as costs associated with finalizing allocations at every decision cycle. Parker (1997) allows robot to explicitly estimate its own as well as performance of the other robots to locally reallocate tasks.

Emergent coordination is another class of techniques in which individual robots coordinate their actions based solely on local sensing information and local interactions (e.g., Rus and Vona, 1999; Martinoli, 1999; Salemi and Shen, 2001). Typically, there is a limited or no direct communication or explicit negotiations among robots. Zhenwang and Gupta (2009) present an adaptive approach to formation problem of a multi-robot system. Specifically, they focus on the communication constraint imposed on a distributed robotic system. They devise a multi-robot system with an adaptive control strategy that maintains the connectivity among the robots throughout the operation. Furthermore, they show this connectivity is uninfluenced with the underlying topology of a specific formation and is achieved with a low cost of the communication among the robotic agents. Dahl et al. (2009) introduce a vacancy chain scheduling technique to achieve an optimal allocation strategy based on the stigmergic effects of robots interactions. However, the dependency of the overall system performance on an emerging strategy is the essential limitation of such approaches. Emergent systems tend not to be amenable to analysis, with their exact behavior difficult, if not impossible, to predict (Gerkey and Mataric, 2004, p. 940).

Parker and Zhang (2008) introduce a consensus-based framework to enable robots to compare their estimates to a threshold while inferring on their respective estimates of a

given task. The consensus is proportional to the size of the robotic team and number of robots that believe the current task is complete. While working on a current task, robots continually estimate the progress of the task. Once individuals infer that current task is complete, they enter the deliberating state. Robots continually communicate with their teammates in the vicinity in a one-by-one basis to inspect their respective beliefs to determine if the current task is completed. If a robot simultaneously receives messages from two or more teammates, the colliding messages are assumed to be lost. After a deliberating robot believes all individuals have come to a common consensus on the completion of a given task, the robot enters the committed state. The committed robots are no longer concern with the opinions of their teammates. However, they exit their committed state and start working on next task if the elapsed time from the last commit-message exceeds a preset limit.

Parker and Zhang (2010) propose a collective unary decision-making mechanism that allows individual robots to update their shared knowledge in a coordinated fashion. The unary decisions are made based on two behavioral states of the robots namely, deliberating state and committed state. Robots monitor the progress of the current task independently to make decisions. Parker and Zhang (2011) examine a biologically inspired collective comparison strategy that allows a swarm of robots to compare alternatives, collectively. The approach relies solely on short range explicit peer-to-peer communication to eliminate any reliance of the system on the stigmergy. The proposed strategy is demonstrated to converge.

1.4 Summary of the Reviewed Research

The review of the literature on multi-robot systems shows that some approaches (e.g., Litus et al., 2009) are strictly domain-specific and are not suitable for different scenarios in which decision-making and coordination are necessary. It is apparent that prioritization and/or predefined robots-to-task assignments are not an effective solution since this strategy is inherently incapable of capturing the dynamical changes that can be incurred in such problem domains. In other words, the fixed nature of prioritized assignments does not allow a readjustment of tasks during the operation.

In contrast, approaches based on mathematical modeling (e.g., Kim and Sugie, 2007; Guo et al., 2010) demand the problem domain to be well-defined in order to be able to derive the equations that model the behavior of the agents. In these approaches complete

information about agents and the operational environment (i.e., type of the terrain, presence or absence of obstacles) are hard-coded into the adapted strategy of the system and consequently, require modifications in response to any changes that are applied to problem domain.

The requirement of *a priori* information in form of some probability distribution over the task space is the main drawback of approaches based on stochastic processes (e.g., Chalkiadakis, 2003). Such *a priori* information varies from equal probability of finding target in all locations of the environment at the commencement of the mission, to uniform probability distribution of presence of the agent over all available locations of the environment during the localization (e.g., Thrun et al., 2001). It is apparent that the availability as well as the accuracy of such information is not warranted. Furthermore, in order for the system to make further decisions over distributions some cues are required (e.g., landmarks that follow a certain trends while localizing). These cues provide agents with the possibility of modifying the given probability distribution and obtaining higher confidence (i.e., *agent's belief*) over probabilities until the goal of the system is achieved (e.g., an agent localizes itself within the field).

In addition to aforementioned issues, some probabilistic approaches (e.g., Chalkiadakis, 2003; Chalkiadakis and Boutilier, 2003) require additional subdivisions of the overall problem into a number of sub-games in order to find an ideal suited strategy at every solution cycle. The divide-and-conquer is an effective approach to finding solutions for large problems. However, the increase in the number of sub-games results in the growth of the complexity of the decision-making and hence the inference of the effective coordination strategy. This is due to the fact that each sub-game adds an additional layer of analysis to obtain the system-level best suited policy.

On the other hand, the market-based approaches are mostly static in finalizing decisions (e.g., Gerkey and Mataric, 2002; Kose et al., 2004). In addition, all tasks are assigned via a central controller and before the commencement of the mission. Single-task assignment is another limitation of market-based strategies (e.g., Lagoudakis et al., 2005; Tovey et al., 2005). These strategies require the entire task space to be divided into a number of subtasks each of which consist of one task while the problem of assignment and coordination is solved individually and sequentially. Some market-based approaches attempt to alleviate the aforementioned shortcomings by either allowing agents to trade their tasks (e.g., Rabideau et al., 1999; Zlot et al., 2002) or by combining the central assignment of multiple tasks,

followed by inter-agent trading of their corresponding assignments (e.g., Dias et al., 2003). However, this solution is infeasible since it duplicates the decision-making and coordination procedures by first making pre-assignments via the introduction of central coordinator and then the trading takes place among agents at the individual level. Therefore, scalability is another drawback of market-based approaches.

Although central schedulers (e.g., Stone, 2007; Kim and Kim, 2002) provide the system with a supervisory ability, the susceptibility of these approaches to fault-tolerance makes them less practical. In general, assignment of the authority of decision-making and coordination for the entire group to one entity, whether a central scheduler or a teammate that plays the role of leader, leaves the system with no decision to be performed in advent of communication loss with that entity. It is a suitable approach for the small- to mid-size robotic teams. However, the overwhelming amount of information to process as number of tasks and/or robotic agents increases makes the idea less appealing. This is due to the computational complexity for the group-level consensus/inference. Considering n agents to perform m tasks, it takes the central scheduler $O(m \times n)$ to calculate possible decisions for individual members of the group, followed by $O(n \times m^2)$ robot-to-task assignment (see Dias et al. (2006) for further complexity analysis of the market-based approaches).

On the other hand, fully distributed strategies (e.g., Karaboga and Akay, 2009) attempt to solve the problem through subdividing the robotic group into smaller teams, each of which propose respective local solutions. The group-level inference is made out of the proposed local solutions, based on some predefined criteria that is a metric (e.g., distance, energy requirement) or a probability distribution function (e.g., mean value of the number of resources that can be covered by each subgroup or individual, deviation of local solutions from some *a priori* benchmark). A pitfall in this approach is local optimality (e.g., optimal solutions that are found in different neighborhoods, formed by robotic subgroups). Increasing the neighborhood size in the local search algorithm or even restarting the algorithm at every system-specific execution or decision cycle helps address the small to medium size problems. A large number of local optima makes the solutions for overcoming local optimality time-consuming and inefficient.

1.5 Operating Assumptions

The underlying architecture of the multi-agent system in this dissertation is a combination of centralized and distributed methodologies. The centralized unit or the mediator coordinates the robotic agents through the mission decomposition (see Chapter 2) and the task allocation (see Chapter 4) processes. On the other hand, the decision-making and execution of the allocated tasks are carried out by the individual robotic agents in a distributed fashion (see Chapter 3). As a result, the individual agents communicate their decisions solely with the mediator and the assignment of tasks is not shared at the group-level. Furthermore, we assume these communications are synchronous and without any further delays. In addition, we assume that information such as the distributions of tasks and the locations of robots are continuously and accurately available to the mediator.

We model our agents as two-wheeled robots. They interact with their surrounding environment based on their respective simulated on-board sensors with 180° field of view and the maximum range of 5 meters. In addition, they perform simple reactive collision avoidance to avoid collision with the obstacles and the robotic agents in their vicinity. Moreover, they navigate with a constant $10\frac{m}{s}$ velocity in case studies presented in Chapters 5 and 6. Their velocity is assumed to vary between 0 and $10\frac{m}{s}$ in Chapter 7.

1.6 Contributions

This thesis contributes to the understanding of intentional cooperation in multi-robot systems. We show that in a complex multi-robot system cooperation is achieved implicitly through the mediation of independent decisions that are made autonomously. We claim that in order for a multi-robot system to accomplish the objective, it is necessary for the individuals to proactively contribute to planning, to incremental refinement, as well as to adaptation at the group-level as the state of the mission progresses.

We show how the decomposition of a mission delegated to a multi-robot system provides the system with capability of distributed decision-making. While formulating decision mechanism, we show how state of every agent with respect to the mission is viewed as two independent internal and external states. Furthermore, we demonstrate the requirement of *a priori* knowledge in decision process is prevented via incorporating a simple sub-ranking mechanism into the external state component of the robotic agent.

While such independent involvement preserves autonomy of the agents, we show the mediation of such independent decisions does not only ensure proper execution of the plan but serves as a basis for evolution of intentional cooperation among individuals.

The contributions of the thesis to the understanding of intentional cooperation in multi-robot systems can be summarized as follows:

1. It develops and analyzes systematic approaches for mission decomposition that are formulated on a robust mathematical basis, yet exhibit high flexibility and extendability as per mission specifications. We introduce the subgrouping (see section 2.1) to reduce the cardinality of the task space. This has a substantial influence on the decision-making and coordination processes (see Chapter 2, Lemma 2). We evaluate the performance of the subgrouping decomposition in contrast to Hungarian algorithm (see Appendix D and Kuhn, 1955) to demonstrate the improvement of the allocation strategy. Furthermore, we demonstrate the ORD (see section 2.2.2) and the LSRD (see Appendix A) techniques utilize the distribution of the robotic agents to transform the decomposition process to an optimization problem. We compare the performance of these strategies with the fixed facility location (see Silverman et al., 2002; Oh and Zelinsky, 2000) and a single dynamic rendezvous location strategy (e.g., Zebrowski et al., 2007). Moreover, we show that the performance of the approach is unaffected by the constraints imposed on the number of attendees of the rendezvous locations (see Chapter 4, Lemma 7 through Lemma 9). In addition, we extend the result in pursuit-evasion research to alleviate the necessary condition of the confinement of the initial location of the intruder within the convex hull of the locations of pursuers (see Jankovic, 1978). We demonstrate that the isogonic decomposition (see section 2.3) achieves this confinement through the computation of a set of virtual goals (see Chapter 2, Definition 1) that are independent of the locations of the pursuers. We also show that the location information of the intruder is sufficient to compute the set of virtual goals.

2. It demonstrates a novel decision mechanism based on subdivision of internal and external states of robot with respect to a delegated mission. This prevents the requirement of *a priori* knowledge for decision-making through a simple opportunistic ranking module. We study the performance of the proposed decision mechanism in contrast to the Bayesian formalism (see Appendix D) to demonstrate the result of decision

process is unaffected by the absence of *a priori* information.

3. It introduces two novel multi-robot coordination strategies capable of preserving group-level optimality of resultant allocations throughout the operation, solely based on the independent decisions of individual agents. We examine the performance of these coordination strategies in comparison to the leader-follower (e.g., Undeger and Polat, 2010), the prioritization, the instantaneous, and the time-extended allocation strategies (see Dias et al., 2006) to demonstrate the optimality of its allocation strategy (see Chapter 4, Lemma 7 through Lemma 9, and Theorem 7).

1.7 Thesis Structure

Chapter 2 represents the mathematical methodologies adapted for the decomposition of a mission. Three decomposition techniques are formulated and analyzed in this chapter. These are *task subgrouping*, *linear*, and *isogonic* decompositions.

Chapter 3 is devoted to the formalization and analysis of individual agent decision engine. While developing individual agent decision engine, we show how the agent states with respect to the mission can be subdivided into two independent *internal* and *external* states. We further demonstrate how the assumption of *a priori* knowledge for decision-making is prevented via the introduction of a novel sub-ranking of agent external state.

Chapter 4 introduces two coordination strategies, namely the *agents votes maximization* (see section 4.2) and the *profile matrix permutations* (see section 4.1). Furthermore, the complexity of aforementioned coordination strategies along with their respective optimality criteria are analyzed in this chapter (see Chapter 4, Lemma 7 through Lemma 9, and Theorem 7).

We illustrate our proposed coordination through the mediation of independent decisions formalization by providing solutions to three non-trivial realistic multi-robot mission scenarios in Chapters 5, 6, and 7, respectively.

In Chapter 5, we demonstrate the applicability of coordination through the mediation of independent decisions in multi-robot dynamic multi-task allocation. These problem domains play a central role in search and rescue, foraging resources as well as environmental catastrophic/hazardous evacuation and exploration scenarios. We show how systematic decomposition, based on the distribution of the subgoals of the overall mission, facilitates the process of decision-making and ranking of the available tasks. This expedites the allocation

procedure in real-time and in an incremental progress of the state of the mission. We show that the combination of subgrouping (see section 2.1) and profile matrix permutations (see section 4.2) enables a multi-robot system to perform a multi-task allocation in a scale that is significantly larger than one-to-one mapping of the robots and the subtasks. We examine the performance of the profile matrix permutations coordination strategy in comparison to the prioritization, the instantaneous, and the time-extended allocation strategies. We consider the elapsed time, the distance traveled and the frequency of the decision cycles to evaluate the performance of these strategies. We use the Hungarian algorithm (see Appendix D and Kuhn, 1955) to coordinate the votes in the instantaneous and the time-extended coordination scenarios.

Chapter 6 studies a variation of multi-facility optimization problem that is useful in a variety of problem domains such as rendezvous, mid- to large-scale complex multi-robot recharging or as a component of various other missions. Throughout the chapter, we demonstrate how the execution of such mission is facilitated via the decomposition of the overall mission into series of subgoals. This benefits from coordination that is inferred from independently conducted decisions. We demonstrate the ORD (see section 2.2.2) and the LSRD (see Appendix A) techniques in conjunction with the agents votes maximization coordination (see section 4.1) strategy utilize the distribution of the robotic agents to transform the decomposition process to an optimization problem. We compare the performance of the agents votes maximization strategy with the fixed facility location (see Silverman et al., 2002; Oh and Zelinsky, 2000) and a single dynamic rendezvous location strategy (e.g., Zebrowski et al., 2007). Moreover, we show that the performance of the approach is unaffected by the constraints imposed on the number of attendees of the rendezvous locations.

Pursuit and evasion scenarios received a special attention in all applications in multi-robot systems. This is due to their inherent complexities and the roles they play in military related missions, border patrol, and other similar domains. Chapter 7 demonstrates the practicality of the proposed methodology in multi-robot single intruder pursuit problem. The main focus of the chapter is concentrated on the decomposition of a pursuit mission that provides the system with a reliable strategy on pursuing and capturing the intruder through coordination that is inferred using the independent decisions of agents. We extend the result in pursuit-evasion research to alleviate the necessary condition of the confinement of the initial location of the intruder within the convex hull of the locations of pursuers (see Jankovic, 1978). We demonstrate that the isogonic decomposition (see section 2.3)

achieves this confinement through the computation of a set of virtual goals (see Chapter 2, Definition 1) that are independent of the locations of the pursuers. We also show that the location information of the intruder is sufficient to compute the set of virtual goals. Furthermore, we study the performance of the agents votes maximization and the profile matrix permutations strategies in contrast to the leader-follower (e.g., Undeger and Polat, 2010), the prioritization, and the probabilistic approaches (see Appendix D).

Chapter 2

Mission Decomposition

In this chapter we present a systematic approach to the decomposition of a mission. In particular, we demonstrate how the distribution of the robotic agents as well as the subtasks provides the system with the opportunity for further subdivision of the overall mission. The decomposition of the task space facilitates the decision-making and the coordination of the task-allocation processes.

A delegated mission to a robotic team is either a composition of several subgoals (e.g., foraging, a rescue mission) or monolithic in nature, but its fulfillment demands the cooperative engagement of several agents (e.g., box-pushing, enclosing an intruder). Mission decomposition is a step towards subdivision of the high level description of a mission into a set of virtual goals such that their incremental executions lead the system to the accomplishment of the mission. Virtual goals are the means through which agents engage with the overall mission. An important aspect of a set of virtual goals $\mathscr{VG}$ is that it forms the common knowledge of the members of the robotic team. More specifically, all robotic agents utilize the same set of virtual goals to make their decisions.

Definition 1 (Virtual Goals)**.** *The set of virtual goals $\mathscr{VG}$ of a robotic team, is a non-empty, finite set of disjoint elements $\rho_j \in \mathscr{VG}$ where every element ρ_j is representative of a subgoal performed by a robotic agent:*

$$\mathscr{VG} \nsubseteq \emptyset \tag{2.1}$$

$$\mathscr{VG} \rightarrow \{1 \dots m\} \;\equiv\; |\mathscr{VG}| = m \tag{2.2}$$

$$\forall \rho_i, \rho_j \in \mathscr{VG}, \rho_i = \rho_j \Leftrightarrow i = j \tag{2.3}$$

A conceptual diagram of the decomposition process is depicted in Figure 2.1. The decomposition process commences with an analysis of the specification of the mission. This analysis provides the system with the necessary information to determine further decomposition steps. For example, the specification declares the mission as a pursuit game, a rendezvous problem or a rescue scenario. Once the nature of the mission is realized, the appropriate steps for the decomposition of the mission are executed. There are three steps involved in the decomposition process:

1. *Task subgrouping*: overall mission is divided into a number of subgroups based on the distribution of the subtasks within the body of the mission. For every subgroup a representative element is calculated. These representatives form the elements of the set of virtual goals (see section 2.1).

2. *Linear decomposition*: the distribution of the robotic agents in their field of operation

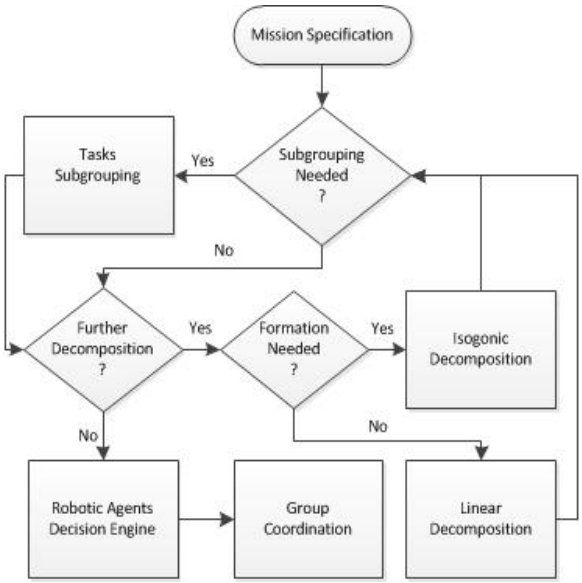

Figure 2.1: The conceptual diagram of decomposition process. The specification of a mission determines whether steps such as subgrouping, isogonic, or linear decompositions are executed. The result of this process is a set of virtual goals that is utilized for the decision-making and coordination.

is utilized to decompose the mission into a set of virtual goals (see section 2.2).

3. *Isogonic decomposition*: the virtual goals are calculated such that their spatial organization reflects a specific formation (see section 2.3).

The remainder of the chapter is organized as follows. Section 2.1 presents the subgrouping process. We demonstrate the decomposition process using the ORD in section 2.2. The virtual goals generation based on final formation of the robotic agents is provided in Section 2.3. The summary and further discussion is provided in section2.4.

2.1 Task Subgrouping

Task subgrouping focuses primarily on the distribution of the subtasks within the body of the mission. This distribution information provides the system with the opportunity to reduce the overall task space into a compact representation where the overall mission is divided into subgroups. The subdivision of the overall mission into a number of subgroups is a step towards facilitating the decision-making and the coordination processes. More specifically, it reduces the decision-making of individual agents to a vote for the representatives of the available subgroups. These votes are in fact an estimation of the robots for all the elements of a given subgroup. This, in turn, transforms the coordination process from one-to-one mapping of agents to individual subtasks to an allocation procedure that takes into account subgroups for allocation. The task subgrouping process exploits the location information of subtasks to generate subgroups. Furthermore, it is important to subdivide the mission into a number of subgroups with an approximately equal number of elements to prevent any uneven distribution of the workload among the robotic agents.

There is a paucity of research on the field of task subgrouping. The adapted strategies are founded mainly on the methodologies that focus on the distributions of labor and not on the subdivision of the task space. Ostergaard et al. (2001) introduce an emerging strategy, entitled the bucket brigading, to partition the task space. In this approach, each robot focuses on a sub-region of the overall task space. Once a resource is located, the robot transports the resource to a neighboring sub-region. The sub-regions are selected such that the direction of the work flow eventually transfers the resources to the main storage area. In other words, robots pass the resources on, region-by-region, towards a base. Shell and Mataric (2006) compare homogeneous foraging and bucket brigading algorithm. The study

emphasizes the importance of the spatial subdivision in reducing inter-agents interference and performance improvement. Lein and Vaughan (2008) extend the result via introduction of a mechanism that adapts the size of the working areas of the robots in response to the interference experienced by the agents. Parker and Zhang (2010) study the performance of a group of robots with a predefined sequence of two mutually exclusive subtasks. A subtask is started only if the preceding subtask is completed. Subtasks are allocated to individual robots and are not shared at a given time. However, this study focuses mainly on the decision-making mechanism where robots are able to collectively estimate if a given subtask is completed. Pini et al. (2011a) partition the task space into two sequentially interdependence subtasks. They are the harvesting of the resources, and the transporting of the harvested resources to the home area. Consequently robots are divided into two teams of harvesting and transporting robots. The subtasks are performed sequentially to complete the global task. Pini et al. (2011b) extend the approach to provide robots with the ability to determine if the partitioning of the delegated task is required.

On the other hand, pattern recognition and other related techniques in AI (see Tou and Gonzalez (1974) for details) are mostly concerned with the correlation of the data based on which clustering of information into distinct sets is delivered. For example, they analyze an image to realize patterns that correspond to fields, roads, boundaries, and so forth. In addition, they sometimes include geometrical measures such as the Euclidean distance to a pivotal or reference point (e.g., using cues of interest to determine if certain data resides below or above a given line). However, these measures are utilized to arrive to a statistical conclusion other than to realize a geometrical correspondence.

Voronoi Diagrams (e.g., Boots et al., 2006; Okabe and Boots, 2000) are also used for subgrouping the task space. They employ a set of pivotal points, referred to as generators, to subdivide the elements of the task space. Kamal et al. (2010) use the locations of the robots as generators to subgroup tasks based on nearest neighbor to the generators. However, the subgrouping is possible only if the robotic agents are within the convex of the task space. This limits the scope of the subgrouping operation and postpones the decision-making and the allocation processes during the period that robots are moving towards the field of operation. Alternatively, Okabe and Suzuki (1997) produce the subgroups using the location information of the subtasks. More specifically, they divide the task space into a number of singleton subsets of one element. It is possible to utilize auxiliary generators to subgroup the task space. The locations of the specific subtasks or a uniformly distributed

set of random locations are examples of such auxiliary generators. These auxiliary generators are introduced within the convex of the subtasks to generate subgroups that are not singleton subsets of the task space. However, this alternative to the choice of generators do not evenly distribute tasks. Depending on the distribution of the subtasks, a subgroup can be overwhelmed with too many elements while another subgroup has a very few or only one member. The increase in the dimension (e.g., multiple runs of the algorithm on the hierarchical results until an even distribution of the assignments is achieved) is a heuristic solution to the aforementioned issue. However, the proportional growth in the complexity of the calculation with the increase of the dimension makes it less practical.

Karavelas (2004) shows that the complexity of the most efficient algorithms for generating Voronoi diagrams is of order of $O(m \log m)$, where m represents the total number of subtasks. It is apparent that the increase in the complexity is proportional to the number of the subgroups when further subgrouping is necessary. For example, if the total number of subgroups is equal to the number of robotic agents n, the increase in the complexity due to the iterative merging and subgrouping of the subgroups is proportional to n (i.e., $O(n[m \log m])$).

We address the above limitations through a subgrouping approach that is based on the percentile values of the distributions of the subtasks. The percentile values are indicators in a given dataset (e.g., location information, delivery time, or the level of acuteness of the subtasks) that represent uniform distributions of elements of the dataset between subsequent percentile values. We subgroup the elements of the task space at every decision cycle. Therefore, the subgrouping process updates the elements of the subgroups dynamically and in conjunction with their distributions at a given decision cycle. This results in the flexibility of the elements of subgroups to vary between consecutive decision cycles.

We use a hypothetical reference point (e.g., origin of the frame of the reference) and sort the subtasks, distance-wise, in ascending order. We further exploit these distances in conjunction with their corresponding percentile values to subdivide the task space into a number of subgroups that have approximately equal number of subtasks. The proposed approach yields the time complexity of $O(m(1 + \log m) + n))$ (see Lemma 1 for the proof), where m and n represent the size of the task space and the number of robotic agents. Furthermore, the subgrouping is achieved in one iteration at a given decision cycle. Therefore, the computational complexity of the proposed approach is unaffected by the size of the task space or the number of the robotic agents. The process takes place in two steps. In the first

step, the task space is divided into a number of subgroups (see section 2.1.1). In the second step, we calculate a representative element for every subgroup (see section 2.1.2).

2.1.1 Subgroup Generation

Consider a scenario where the overall mission delegated to a robotic team consists of a number of subtasks, with robots equally capable of performing each task. We refer to the set $\mathscr{T}$ that comprises the location information of all these subtasks by the term *task space*.

Definition 2 (Task Space)**.** *Task space $\mathscr{T}$ of a robotic team is a set where every element $\tau_i \in \mathscr{T}$ is representative of the location information of a subtask of the overall delegated mission.*

It is apparent that by the definition:

$$\mathscr{T} \nsubseteq \emptyset \tag{2.4}$$

$$1 \leq |\mathscr{T}| \leq m \tag{2.5}$$

$$\forall \tau_i,\ \tau_j \in \mathscr{T}, \tau_i = \tau_j \Leftrightarrow i = j \tag{2.6}$$

That is $\mathscr{T}$ is a non-empty, finite set of disjoint elements. Definition 2.6 does not restrict our notion of subtasks to stationary objects. More specifically, we generalize the idea to the subtasks that exhibit dynamic behaviors and change their locations in space.

Given the task space $\mathscr{T}$, we are interested in the subgrouping of elements of $\mathscr{T}$ such that an approximately even allocation of subtasks to the robotic agents is possible. These subgroups reflect the distribution of the subtasks in the field and are independent of the locations of the robotic agents. Furthermore, with n of robots, it is desirable to decompose the mission into n subgroups where each subgroup is associated with one member of the robotic team.

We subdivide the elements of $\mathscr{T}$ using percentile values of the distribution of the subtasks. This process subdivides the task space into a number of subgroups that are $\frac{100}{n}$ percentiles apart. For instance, if $n = 5$, the subgroups are the subsets of $\mathscr{T}$ that fall between every 20^{th} percentile with respect to the overall distribution of the elements of the task space. The P^{th} percentile ($0 \leq P \leq 100$) of a set of values that are sorted in ascending

order is calculated as (see Gravetter and Wallnau (2008) for details):

$$\varrho = round(\frac{P}{100} \times m + 0.5) \tag{2.7}$$

where m is the total number of the elements in the set and the function *round* returns the nearest integer to ϱ. For example, if $\{10, 12, 14, 28, 62, 80\}$ represents the set of sorted elements in ascending order, the 20^{th} percentile is the element at the position $\varrho = \frac{20}{100} \times 6 + 0.5 = 1.7 \approx 2$. This is the 2^{nd} element of the set, 12.

However, we need to sort the elements of the task space $\mathscr{T}$ before the application of the percentile values for subgrouping. We sort $\mathscr{T}$ using the corresponding distances of its individual members $\tau_i \in \mathscr{T}$ to a specific reference point $\mathscr{O}$. This reference point is an arbitrary point in the field of operation (e.g., the origin of the frame of the reference). Furthermore, the location of the reference point does not influence the subgrouping procedure (i.e., biased subdivision of $\mathscr{T}$) if the same reference point is used to calculate the distances of the individual elements of the task space $\tau_i \in \mathscr{T}$.

We calculate the array $\mathscr{D}$ that contains the distances of the subtasks $\tau_i \in \mathscr{T}$ to the reference point $\mathscr{O}$ and use its entries to obtain the reordered array of indices of the subtasks, denoted by χ. The rearrangements of the indices of the subtasks in χ reflect the sorting of the subtasks to the reference point $\mathscr{O}$ in ascending order. For example, if the 4^{th} element of the task space $\tau_4 \in \mathscr{T}$ has the shortest distance to $\mathscr{O}$ among all $\tau_i \in \mathscr{T}$, the index 4 forms the 1^{st} entry of χ.

Algorithm 1 shows the process of rearrangements of the indices of $\mathscr{T}$ in χ. The process begins with computing $\mathscr{D}$ and χ. Next, the entries of these arrays are reordered, distance-wise, in an ascending order. Every time an element of the distance array $\mathscr{D}$ is swapped, its corresponding index value in χ is relocated to comply with the new positioning of the subtasks that are sorted in ascending order. In Algorithm 1, the function $dist(x, y)$ returns the Euclidean distance between its two arguments.

The sorted array of indices χ is utilized (see Algorithm 2) to subdivide the task space

Algorithm 1: Sorting of the indices of $\tau_i \in \mathscr{T}$ in ascending order with respect to their corresponding distances to reference point $\mathscr{O}$.

Data: $\mathscr{T}$, Task space of the mission to be decomposed.
Data: m, Size of task space i.e $|\mathscr{T}| = m$.
begin
  for $i = 1 : m$ **do**
    $\mathscr{D}[i] = dist(\mathscr{O}, \tau_i)$;
    $\chi(i) = i$;
  Sort the elements of $\mathscr{D}$ in ascending order and reflect its ordering in χ;

into a number of subgroups $\mathscr{SG}$ such that:

$$\mathscr{SG}_i \subseteq \mathscr{T} \tag{2.8}$$

$$\mathscr{T} = \bigcup_{i=1}^{n} \mathscr{SG}_i \tag{2.9}$$

$$\bigcap \mathscr{SG}_i = \emptyset \tag{2.10}$$

where $\mathscr{T}$, n, and $\mathscr{SG}_i$ represent the task space, the total number of the robotic agents, and the i^{th} subgroup, respectively. Equation (2.10) specifies that the subgroups are disjoint subsets of $\mathscr{T}$. It is apparent that the union of the subgroups is the original task space $\mathscr{T}$.

Algorithm 2 demonstrates the process of the subdivision of $\mathscr{T}$ into n subgroups, where n represents the total number of the robotic agents. The parameters χ, n, and $m = |\mathscr{T}|$ (i.e., the size of the task space) are the inputs to the algorithm. It first calculates the range of the percentile values using n. The algorithm tracks the previously calculated percentile value in variable ϱ_{prev}. Moreover, the algorithm uses the *floor* function $\lfloor \varrho \rfloor$ to obtain the closest lower-bound integer to the calculated percentile value.[1] This avoids overlapping elements among different subgroups where subtasks fall onto two or more subgroups.

The algorithm next uses ϱ_{prev} and ϱ (the latter holds the currently calculated percentile location) to extract the indices of subtasks $\tau_i \in \mathscr{T}$ from the entries of χ. These indices are saved in the matrix *SubGroups*. The row entries of the matrix *SubGroups* correspond to different subgroups $\mathscr{SG}_j \subseteq \mathscr{T}$. On the other hand, the column entries of the matrix are the indices of subtasks $\tau_i \in \mathscr{T}$ that fall onto the subgroup $\mathscr{SG}_j$.

[1] As shown in equation (2.7), the conventional technique uses the *round* function to obtain the nearest integer instead.

Algorithm 2: Percentile value subgrouping of task space $\mathscr{T}$.

Data: χ, Distance-wise sorted array of indices of $\tau_i \in \mathscr{T}$.
Data: n, Number of robotic agents.
Data: m, Size of task space i.e $|\mathscr{T}| = m$.
begin
  $range \leftarrow \frac{100}{n}$;
  $Percentile \leftarrow range$;
  $\varrho_{prev} \leftarrow 1$;
  $j \leftarrow 1$;
  while $n \geq 1$ **do**
    $\varrho = \lfloor \frac{Percentile}{100} \times m + 0.5 \rfloor$;
    if $\varrho > m$ **then**
      $\varrho \leftarrow m$;
    for $i = \varrho_{prev} : \varrho$ **do**
      $SubGroup[j][i] \leftarrow \chi[i]$;
    $\varrho_{prev} \leftarrow \varrho + 1$;
    $Percentile \leftarrow Percentile + range$;
    $j \leftarrow j + 1$;
    $n \leftarrow n - 1$;
  return $SubGroup$;

The algorithm ensures that the value of ϱ does not exceed the size of $\mathscr{T}$ through the if block. The value of the *Percentile* variable is updated at every iteration to reflect the location of the next percentile value in χ. The iterations are repeated until the mission is divided into n subgroups (the outer *while* loop) where n represents the total number of the robotic agents.

Lemma 1 (Subgroups Computation Complexity)**.** *Given a task space $\mathscr{T}$ with the cardinality $|\mathscr{T}| = m$ and a team of n robotic agents, $n \ll m$, it takes no longer than $O(m(1+\log m)+n))$ to divide $\mathscr{T}$ into n subgroups.*

Proof. Let m represent the total number of the elements of $\mathscr{T}$. It takes the time $O(m)$ to calculate the distances of the subtasks $\tau_j \in \mathscr{T}$ to the reference point $\mathscr{O}$. Sorting these distances in ascending order is done in $O(m \log m)$. Every percentile value ϱ is located in $O(1)$. Therefore, to subgroup $\mathscr{T}$ into n subgroups is $O(n)$. This results in:

$$O(m) + O(m \log m) + O(n) = \tag{2.11}$$

$$O(m(1 + \log m) + n)) \tag{2.12}$$

□

2.1.2 Subgroup Representative Calculation

Earlier in this Chapter we claimed that the subdivision of the overall mission into a number of subgroups is a step towards facilitating the decision-making and the coordination processes. We noted that the subdivision of the task space reduces the decision-making of the individual agents to vote for the representatives of the available subgroups instead of individual subtasks of the overall task space. We define the representative of a subgroup as:

Definition 3 (Subgroup representative)**.** *Representative $\hat{\tau}_i$ of a subgroup $\mathscr{SG}_i \subseteq \mathscr{T}$ is an auxiliary element of the subgroup with its location, at every instance of time, corresponds to the center of the mass of the elements $\tau_j \in \mathscr{SG}_i,\ j = 1 \ldots |\mathscr{SG}_i|$.*

We calculate the representative of a subgroup using the location information of the elements of the subgroup.[2] An interesting property of the representatives is that their locations within the body of their corresponding subgroups minimize the cumulative sum of the distances of the elements of their subgroups (see Boltyanski et al. (1999) for the proof):

$$\hat{\tau}_i = argmin \sum_{\tau_j \in \mathscr{SG}_i} w_j \|p - \tau_j\|, \quad \mathscr{SG}_i \subseteq \mathscr{T} \tag{2.13}$$

We use equation 2.13 to calculate the representative of a subgroup at every execution cycle. The representative $\hat{\tau}_i$ of the subgroup $\mathscr{SG}_i$ is calculated as:

$$\hat{\tau}_i = argmin \sum_{j=1}^{s} w_j \|p - \tau_j\| = \frac{1}{s} \sum_{j=1}^{s} w_j \tau_j,\ \forall \tau_j \in \mathscr{SG}_i,\ |\mathscr{SG}_i| = s \tag{2.14}$$

where s represents the cardinality of subgroup $\mathscr{SG}_i$ and w_j is the weight associated with j^{th} member of $\mathscr{SG}_i$. The weight factor w_j controls the location of $\hat{\tau}_i$ to be closer or farther from specific members of $\mathscr{SG}_i$, if necessary. We assign $w_j = 1$ to calculate the representatives of the subgroups. More specifically, we treat all the elements of the subgroups equally during the calculations of their representatives.

Lemma 2 (Reduction of the Complexity)**.** *The subgrouping process reduces the complexity of any further computation performed by a multi-robot system, proportional to the size of the robotic team.*

[2]These representatives are not the actual subtasks. However, it is possible that the location of a representative coincides with a subtask at a given execution cycle.

Proof. Let m and n, $n \ll m$, represent the size of the task space and the total number of the robotic agents, respectively. The subgrouping process generates a number of subgroups that equals n. Furthermore, it represents every subgroup via an auxiliary element $\hat{\tau}_i \in \mathscr{SG}_i \subseteq \mathscr{T}$. This implies that any further computation on the task space is performed on $\frac{m}{n}$ representatives. This reduces every $O(m)$ to $O(\frac{m}{n})$. □

2.2 Linear Decomposition

In certain problem domains such as facility location (see Calamai and Conn (1980); Schlude (2003) for examples),[3] the relocation of robotic agents is a factor that significantly influences the outcome of the strategy adapted to perform a delegated mission. Although finding a closed-form solution for this class of problems is intractable, mission decomposition provides a means to formally study and analyze the proposed solution in conjunction with the overall performance of the system. In this respect, mission decomposition is expressed as an optimization problem where an optimal solution to certain instances of the problem domain is determined.

Definition 4 (Optimal Relocation Strategy)**.** *Given the location information of a team of robotic agents $r_i \in \mathbb{R}^d$ $(d \geq 2)$ and the relocation cost function $C_i : \mathbb{R}^d \to \mathbb{R}$, i=1 ... n, an optimal relocation strategy decomposes the mission into a set of virtual goals $\rho_j \in \mathscr{VG}$ such that*

$$\min \sum_{i=1}^{n} C_i(r_i, \rho_i), \ \rho_i \in \mathscr{VG} \tag{2.15}$$

is satisfied.

In Definition 4, r_i and ρ_i are the current location of the i^{th} robotic agent, and the location of the virtual goal assigned to the robot after the decomposition of the mission. The function $C_i(r_i, \rho_i)$ returns the cost of the relocation of the i^{th} agent from its current position r_i to the location of the virtual goal $\rho_j \in \mathscr{VG}$:

$$C_i(r_i, \rho_i) = w_i \|\rho_i - r_i\|, \quad i = 1 \ldots n, \ \rho_i \in \mathscr{VG} \tag{2.16}$$

The remainder of the section 2.2 demonstrates the applicability of the regression analysis

[3]Fermat-Steiner problem, Weber problem, single facility location problem, and the generalized Fermat-Torricelli problem are also used synonymously to refer to this class of optimization problem.

(see Appendix A for a brief introduction) to address the aforementioned class of problems. Specifically, we examine the practicality of the least squares and the orthogonal least absolute values (ORLAV) regressions. Mission decomposition based on least squares regression (referred to as *LSRD* for least-square regression decomposition, hereafter) is presented in Appendix A.2. We discuss the limitations of the *LSRD* in section 2.2.1. Next we address these shortcomings through a reformulation of the mission decomposition using an ORLAV-based algorithm (referred to as *ORD* for orthogonal regression decomposition, hereafter). We further investigate the properties attributed to the proposed algorithm in section 2.2.3.

2.2.1 Preliminary: From LSRD to ORD

One of the assumptions in fitting the least squares regression analysis (see Appendix A) is the dependency of the coordinate information of the robotic agents. In other words, the coordinates information of the robotic agents should exhibit an explanatory-response relationship (see Appendix A for further explanation). Furthermore, least squares regression requires that the behavior of one of the coordinates to be predictable by and hence a function of another coordinate. For example, y-coordinate information is a function of x-coordinate information and its relocation is determined by the x-coordinate information. This assumption requires the predefinition of the dependency among the (x, y) coordinates of the individual robotic agents. It is apparent that such an assumption is unrealistic and is not warranted.

In addition, least squares regression assumes that the predictor or the independent variable (see Appendix A for the explanation) is free of error and confines all the errors to the dependent or the response variables (see Amari and Kawanabe (2002) for further justification). Another limitation in using least squares linear regression is when outliers (i.e., robots that are located in farther distances from the rest of the group) are present. Since the objective function in least squares evaluates squares of vertical distances to the hyperplane (see Bargiela and Hartley (1993) for explanation) location information of these robotic agents are unduly heavily weighted.

To overcome these limitations, we propose the orthogonal regression decomposition (*ORD*) using orthogonal least absolute values (ORLAV).[4] Some of the advantages of using

[4]The technique is also referred to as Euclidean minimum sum of absolute errors (EMSAE), Euclidean regression (ER) and total least squares (TLS), interchangeably in the literature.

ORLAV are:

1. No choice of dependent and independent variables is required: The ORLAV obtains the estimators of the orthogonal regression through minimization of the orthogonal distances of location information of the robotic agents to the fitted hyperplane. Therefore, it is applicable for situations where the dependent and the independent variables are not predetermined.

2. It is resistant to the outliers: The ORLAV minimizes the sum of the orthogonal distances of the locations of the individual robot to the fitted hyperplane. This is different from the minimization of the sum of the squares of vertical distances in least squares linear regression (see Melloy and Cavalier (1991) for further explanation).

3. It is ideally suited for situations where the data is corrupted by noise (see Amari and Kawanabe (2002) for further explanation).

2.2.2 Orthogonal Regression Decomposition (ORD)

ORD mission decomposition is based on finding a *1-line median* of the location information of the robotic agents.[5] It exploits the location information of the robotic agents to find a line that minimizes the cumulative sum of the orthogonal distances of the robotic agents to the fitted line. We compute this line to satisfy the necessary condition of spreading the location information of the robotic agents into two sets with approximately the same weights. More specifically, if $W = \sum_{i=1}^{n} w_i$, $T^+ = \{i : ax_i + by_i > -c\}$, and $T^- = \{i : ax_i + by_i < -c\}$ represent the total weights of the robotic agents, and the sets that comprise the robotic agents above and below the fitted line, respectively, we compute the line to satisfy the inequality (see Megiddo and Tamir, 1983, p.207):

$$\sum_{i \in T^+} w_i, \sum_{i \in T^-} w_i \leq \frac{1}{2} W \qquad i=1 \ldots n \tag{2.17}$$

If the equation of the candidate line is expressed as:

$$ax + by + c = 0 \tag{2.18}$$

[5] In location theory the solution to the 2-dimensional problem of finding access to a linear resource (e.g., a highway or a utility) is referred to as 1-line median.

we calculate the values for the parameters a, b and c in equation (2.18) to minimize the weighted sum of the orthogonal distances of the locations of the robotic agents from the line. This line always passes through the locations of at least two of the robotic agents (see Lemma 5). Therefore, the endpoints of the candidate line are always known. Furthermore, equation (2.17) indicates that the candidate line divides the robotic agents into two sets of agents that are above and below the line. Hence, the weighted sum of the orthogonal distances is expressed by the equation:

$$(a^2+b^2)^{\frac{-1}{2}}[(\sum_{i\in T^+} w_i x_i - \sum_{i\in T^-} w_i x_i)a + (\sum_{i\in T^+} w_i y_i - \sum_{i\in T^-} w_i y_i)b + (\sum_{i\in T^+} w_i - \sum_{i\in T^-} w_i)c] \tag{2.19}$$

We use the location information of the robotic agents that coincide with the line, along with the equations (2.17) and (2.19) to calculate the parameters a, b, and c. Alogrithm 3 elaborates the process. The algorithm exploits the location information of the pairs of the robotic agents at every iteration and computes the parameters a, b, and c. It tracks the calculated values in S. In the next step, it utilizes S to select the combination of the parameters that minimizes the equation (2.19).

We use the parameters a, b, and c calculated by Algorithm 3 to compute the virtual

Algorithm 3: The *ORD* mission decomposition

Data: (r_i, r_j) Locations of the pairs of robotic agents 'for iterative line generation.
begin
  $i \leftarrow 1$;
  $k \leftarrow 1$;
  while $i \geq n-1$ **do**
    for $j = i+1 : n$ **do**
      find a_k, b_k and c_k for the line $a_k x + b_k y + c_k = 0$ that passes through (r_i, r_j)
      for $m = 1 : n$ **do**
        if $a_k x_m + b_k y_m < -c_k$ **then**
          $T^+ \leftarrow \{m\}$
          else
            $T^- \leftarrow \{m\}$
      $S \leftarrow \{(a_k, b_k, c_k)\}$
      $k \leftarrow k+1$
    $i \leftarrow i+1$;newline
  return $s \in S$ that satisfies
  $\min[(a_i^2+b_i^2)^{\frac{-1}{2}}[(\sum_{i\in T^+} w_i x_i - \sum_{i\in T^-} w_i x_i)a_i + (\sum_{i\in T^+} w_i y_i - \sum_{i\in T^-} w_i y_i)b_i +$
  $(\sum_{i\in T^+ w_i} + (\sum i \in T^+ w_i - \sum i \in T^- w_i)]]$

goals based on the location information of the robotic agents. In Figure 2.2, the endpoints of the calculated line are (x_1, y_1) and (x_2, y_2). We utilize the coordinates of these endpoints along with the location information of the i^{th} robot (x_{r_i}, y_{r_i}) and calculate the virtual goal $\rho_i \in \mathscr{VG}$ as:

$$x_{\rho_i} = \frac{m^2 x_1 - m y_1 + m y_{r_i} + x_{r_i}}{1 + m^2} \tag{2.20}$$

$$y_{\rho_i} = \frac{(m-1)x_1 + m^2 y_{r_i} + m x_{r_i} + y_1}{1 + m^2} \tag{2.21}$$

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{2.22}$$

2.2.3 Formal Analysis

The *ORD* decomposes the mission through the minimization of the sum of the orthogonal distances of the robots from the collinearly generated virtual goals. It differs from the *LSRD* decomposition (see Appendix A) that minimizes the sum of the squared vertical distances of the robots from the candidate line. This benefits a multi-robot system in several fundamental ways.

Lemma 3 (Collinear Case)**.** *The weighted sum of Euclidean distances of the collinear robots to the virtual goals is always zero.*

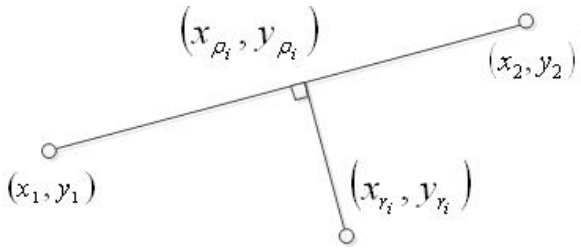

Figure 2.2: The location of virtual goal ρ_i with respect to the calculated line and i^{th} robotic agent.

Proof. Referring to Figure 2.3, the current locations of the collinear robots and their corresponding virtual goals coincide. Hence:

$$\sum_{i=1}^{n} w_i \|\rho_i - r_i\| = 0,\ \rho_i \in \mathscr{VG} \tag{2.23}$$

□

Lemma 4 (Special Case)**.** *In an obstacle-free planar environment, the weighted sum of Euclidean distances of the robotic agents to their corresponding virtual goals is less than the sum of their distances to any arbitrary location along the line that connects the virtual goals.*

Proof. Referring to Figure 2.4, let L be the line that connects the virtual goals $\rho_i \in \mathscr{VG}$. Let f be the arbitrary point along L. In $\triangle r_i \rho_i f$:

$$w_i \|\rho_i - r_i\| \leq w_i \|f - r_i\|,\ \rho_i \in (VG) \tag{2.24}$$

where:
$w_i\|f - r_i\|$ is the weighted distance of i^{th} robot to the arbitrary point f along L.
$w_i\|\rho_i - r_i\|$ is the weighted Euclidean distance of i^{th} robot to the virtual goal ρ_i.

This implies that:

$$\sum_{i=1}^{n} w_i \|\rho_i - r_i\| \leq \sum_{i=1}^{n} w_i \|f - r_i\|,\ \rho_i \in \mathscr{VG} \tag{2.25}$$

□

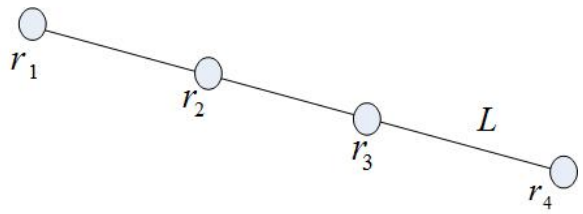

Figure 2.3: The collinear Robots.

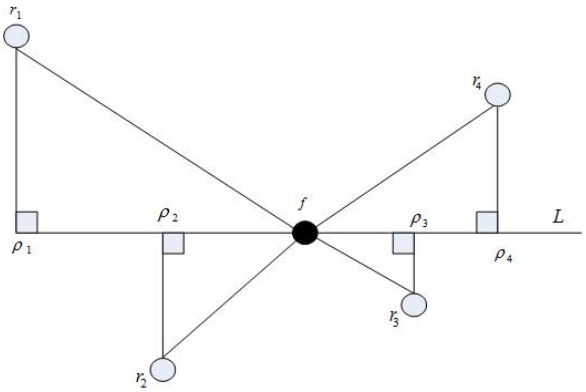

Figure 2.4: Robots r_i and their corresponding virtual goals $\rho_i \in \mathscr{VG}$. The arbitrary point along the line L is labeled as f.

Theorem 1 (General Case)**.** *In an obstacle-free planar environment, the weighted sum of Euclidean distances of the robots to their corresponding virtual goals is less than the sum of their distances to any arbitrary location on the plane.*

Proof. In Figure 2.5, let L be the line that connects virtual goals and f be any arbitrary location within the field. We draw a line $L^{'}$ that passes through f and is parallel to L. Using Lemma 4 and considering $\triangle r_i \rho_i' f$:

$$\sum_{i=1}^{n} w_i \|\rho_i' - r_i\| \leq \sum_{i=1}^{n} w_i \|f - r_i\| \tag{2.26}$$

where
$w_i\|f - r_i\|$ is the weighted distance of i^{th} robot to the arbitrary point *f*.
$w_i\|\rho_i' - r_i\|$ is the weighted Euclidean distance of i^{th} robot to the virtual goal ρ_i.
However, L is the line that connects virtual goals $\rho_i \in \mathscr{VG}$. Therefore, the sum of the orthogonal distances of the robots to L is minimized:

$$\sum_{i=1}^{n} w_i \|\rho_i - r_i\| \leq \sum_{i=1}^{n} w_i \|\rho_i' - r_i\| \tag{2.27}$$

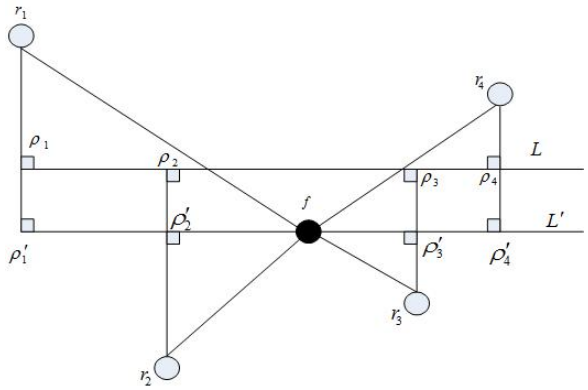

Figure 2.5: Robots r_i and their corresponding virtual goals $\rho_i \in \mathscr{VG}$. The arbitrary point is labeled as f. L and L' are the lines that connect virtual goals, and the line that passes through the arbitrary point f.

Using equation (2.26) and equation (2.27), we get:

$$\sum_{i=1}^{n} w_i \|\rho_i - r_i\| \leq \sum_{i=1}^{n} w_i \|f - r_i\| \tag{2.28}$$

□

Theorem 1 holds true when the location information of the arbitrary point corresponds to the Fermat-Torricelli point of the locations of the robotic agents (see Corollary 1 below). Boltyanski et al. (1999) generalizes the concept of the Fermat problem to find a point $x_0 \in \mathbb{R}$ for a given set of points $p_1 \dots p_m \in \mathbb{R},\ n \geq 1$, that minimizes the function (ibid., p. 236):

$$\mathscr{F}(x) = \sum_{i=1}^{n} w_i \|p_i - x\| \quad x \in \mathbb{R}^d \quad (d \geq 1) \tag{2.29}$$

They further prove that the function $\mathscr{F}$ has the following properties.

Claim 1. *The function $\mathscr{F}$ is convex* (for the proof see Boltyanski et al., 1999, p. 239).

Claim 2. *The function $\mathscr{F}$ is a strictly convex function if and only if the points $p_1, \dots, p_m$ are not collinear* (for the proof see Boltyanski et al., 1999, p. 240).

Furthermore, Meddigo and Tamir prove that the line in the equation (2.18) has the following property if the parameters of this line satisfy the equation (2.19):

Lemma 5. *Relative to the Euclidean distance there exists a 1-line median which contains at least two points from the set* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ (for the proof see Megiddo and Tamir, 1983, p. 207).

We utilize the Claim 1 and the Claim 2 along with the Lemma 5 to generalize the result of Theorem 1 where the arbitrary point is the Fermat point of the location information of the robotic agents.

Corollary 1 (Generalization on the Fermat-Torricelli point)**.** *In an obstacle-free planar environment, the weighted sum of Euclidean distances of the robots to their corresponding virtual goals is less than the sum of their distances to the Fermat-Torricelli point of their location information.*

Proof. We consider the following two cases:

1. *The Fermat point does not coincide with location information of a robotic agent.*
   It is apparent that a non-coincidental Fermat point is analogous to the scenario depicted in Figure 2.5. This implies that the result of Theorem 1 holds true for the Fermat-Torricelli point.

2. *The Fermat point coincides with the location of one of the robots.*
   Figure 2.6 shows this scenario where the Fermat point f coincides with the robotic agent r_3. Claim 1 and Claim 2 ensure that the location of r_3 is within the convex hull of the rest of the robotic agents. Using these claims in conjunction with Theorem 1, we have:
   $$\sum_{i=1}^{n-\{3\}} w_i \|\rho_i - r_i\| \leq \sum_{i=1}^{n-\{3\}} w_i \|r_3 - r_i\|, \ \rho_i \in \mathscr{VG} \tag{2.30}$$
   where n is the number of the robotic agents and $n - \{3\}$ is all the robots except r_3. To generalize the result of the Theorem 1, we prove that the cost of the relocation of r_3 to its virtual goal ρ_3 is less than the cost of the relocations of the robotic agents that are along the line that connects the virtual goals (see Lemma 5).
   In Figure 2.6, let L be the line that connects the virtual goals $\rho_j \in \mathscr{VG}$. Let $L^{'}$ represent the line that passes through f and that is parallel to L. In addition, let r_1 and r_2 represent the robotic agents that are along the line L. Therefore, the sum of

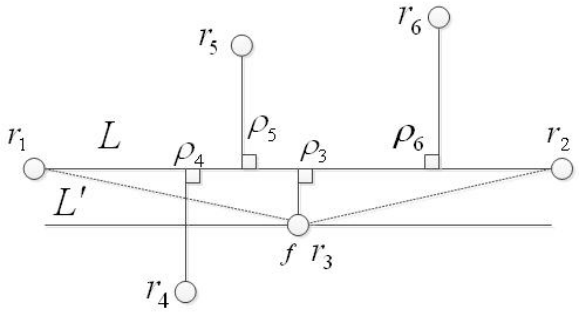

Figure 2.6: Robots r_i and their corresponding virtual goals. The location of the Fermat-Torricelli point f is coincidental with r_3.

the orthogonal distances of r_1 and r_2 to L is zero (see Lemma 3). Hence:

$$\sum_{i=1}^{2} w_i \|\rho_i - r_i\| = 0,\ \rho_i \in \mathscr{V}\mathscr{G} \tag{2.31}$$

Furthermore, in the triangle $\triangle R_i \rho_3 f$ where r_3 and f are coincidental, we have:

$$w_3\|\rho_3 - r_3\| \leq w_1\|f - r_1\| \tag{2.32}$$

$$w_3\|\rho_3 - r_3\| \leq w_2\|f - r_2\| \tag{2.33}$$

$$\Rightarrow w_3\|\rho_3 - r_3\| \leq \sum_{i=1}^{2} w_i\|f - r_i\| \tag{2.34}$$

Equation (2.34) states that the sum of orthogonal distances of the robotic agents r_1 and r_2 is greater than the cost of relocation of the robotic agent r_3. This results in:

$$\begin{aligned} &\sum_{i=1}^{n-\{3\}} w_i\|\rho_i - r_i\| + w_3\|r_3 - \rho_3\| \\ &\leq \sum_{i=1}^{n} w_i\|f - r_i\| + \sum_{i=1}^{2} w_i\|f - r_i\| \\ \Longrightarrow &\sum_{i=1}^{n} w_i\|\rho_i - r_i\| \leq \sum_{i=1}^{n} w_i|f - r_i\| \end{aligned} \tag{2.35}$$

□

2.3 Isogonic Decomposition

Some problem domains such as the pursuit-game, the formation, and the large-object transportation require system to determine the positioning of the individual agents with respect to the other robots within a certain approximation. The underlying structure of this configuration of agents provides a foundation to determine their future displacement and relocation. In other words, it is possible to adapt a top-down approach to the formulation of the final desirable configuration to decompose a mission to a set of virtual goals. Figure 2.7 depicts a scenario where a mission is decomposed to a set of virtual goals $\rho_i \in \mathscr{VG}$ that forms the vertices and the first isogonic point of an isosceles triangle.[6] This isogonic point resides on the intersection of the lines that connect the vertices of the three equilateral triangles that are formed out of the sides of the given triangle to the vertex that is in the opposite side of the given triangle.

As shown through the following Theorem, an interesting property attributed to an isosceles triangle is the alignment of its isogonic point with its leading vertex.

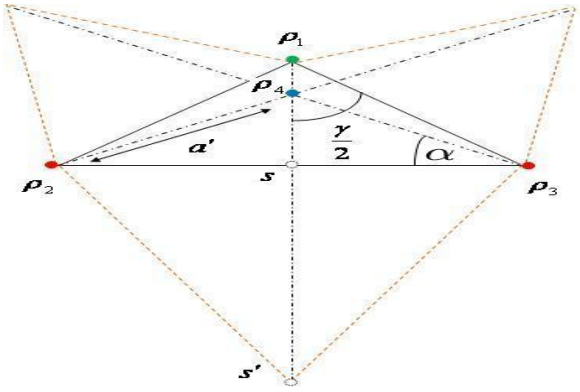

Figure 2.7: The first isogonic formation of the virtual goals.

[6]The isogonic point of a triangle minimizes the cumulative sum of distances of the vertices of the triangle. ρ_4 is the isogonic point of $\triangle\rho_2\rho_1\rho_3$ in Figure 2.7. There are two isogonic points associated with every triangle. Appendix B provides further details on the formulation of this decomposition using the second isogonic point.

Theorem 2 (Isogonic Alignment)**.** *The isogonic point of an isosceles triangle is always aligned with its leading vertex.*

Proof. In the equilateral triangle $\triangle\rho_2 s'\rho_3$ of Figure 2.7, we have:

$$\|\rho_2 s'\| = \|\rho_3 s'\| \;\; \& \;\; ss' \perp \rho_2\rho_3 \Rightarrow \|\rho_2 s\| = \|\rho_3 s\| \tag{2.36}$$

Similarly, in the triangle $\triangle\rho_2\rho_1\rho_3$ of Figure 2.7, we get:

$$\|\rho_2\rho_1\| = \|\rho_3\rho_1\| \;\; \& \;\; \rho_1 S \perp \rho_2\rho_3 \Rightarrow \|\rho_2 s\| = \|\rho_3 s\| \tag{2.37}$$

Equations (2.36) and (2.37) imply that $\rho_1 s$ and ss' are aligned. Therefore:

$$\rho_1 s' \perp \rho_2\rho_3 \;\; \& \;\; \|\rho_2 s\| = \|\rho_3 s\| \tag{2.38}$$

□

Corollary 2. *The isogonic point of an isosceles triangle is always in equal distances from its two side vertices.*

Proof. The proof of this Corollary is apparent in equation 2.38. □

Theorem 2 and Corollary 2 demonstrate that the location of ρ_4 is well-defined with regards to the locations of ρ_1, ρ_2, and ρ_3 (see Figure 2.7). This reduces the amount of information required to decompose a mission into a set of virtual goals. More specifically, we only need to assume the initial location information of one of the virtual goals in order to generate the entire set $\mathscr{VG}$. We assume that the initial location information of ρ_1 is known to compute the set of virtual goals. However, there is no restriction on the choice of this virtual goal.

We exploit the location information of ρ_1 and compute the set of virtual goals. In Figure 2.7, locations of ρ_2 and ρ_3 are computed as:

$$\rho_2 = \begin{bmatrix} x_{\rho_1} - (\|\rho_3\rho_1\| \times sin(\frac{\lambda}{2})) \\ y_{\rho_1} - (\|\rho_3\rho_1\| \times cos(\frac{\lambda}{2})) \end{bmatrix} \tag{2.39}$$

$$\rho_3 = \begin{bmatrix} x_{\rho_1} + (\|\rho_3\rho_1\| \times sin(\frac{\lambda}{2})) \\ y_{\rho_1} - (\|\rho_3\rho_1\| \times cos(\frac{\lambda}{2})) \end{bmatrix} \tag{2.40}$$

where:

$$\|\rho_1 s\| = \|\rho_1\rho_2\| \cos(\frac{\gamma}{2}) = \|\rho_1\rho_3\| \cos(\frac{\gamma}{2}) \tag{2.41}$$

$$\|\rho_2 s\| = \|\rho_1\rho_2\| \sin(\frac{\gamma}{2}) = \|\rho_3 s\| = \|\rho_1\rho_3\| \sin(\frac{\gamma}{2}) \tag{2.42}$$

We utilize the location information of these virtual goals to confine the location of the virtual goal ρ_4 within the convex hull of the $\triangle\rho_2\rho_1\rho_3$. This constraint on the location of the virtual goal ρ_4 is satisfied if and only if $\angle\rho_2\rho_1\rho_3 < 120^\circ$ as shown in the following Theorem.

Theorem 3 (Coincidental Case)**.** *ρ_1 and ρ_4 coincide if $\angle\rho_2\rho_1\rho_3 \geq 120^\circ$.*

Proof. Let ρ_4 represent the isogonic point of the isosceles triangle $\triangle\rho_2\rho_1\rho_3$ where $\angle\rho_2\rho_1\rho_3 \geq 120^\circ$, $\|\rho_2\rho_1\| = \|\rho_3\rho_1\|$ (see Figure B.1). Let:

$$\frac{\rho_4\rho_2}{\|\rho_4\rho_2\|} + \frac{\rho_4\rho_3}{\|\rho_4\rho_3\|} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \tag{2.43}$$

Furthermore, ρ_4 is the isogonic point of $\triangle\rho_2\rho_1\rho_3$ if ρ_4 satisfies (Kupitz and Martini, 1997, p. 58):

$$\frac{\rho_4\rho_2}{\|\rho_4\rho_2\|} + \frac{\rho_4\rho_3}{\|\rho_4\rho_3\|} + \frac{\rho_4\rho_1}{\|\rho_4\rho_1\|} = 0 \tag{2.44}$$

Substituting equation (2.43) in equation (2.44), we get:

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} + \frac{\rho_4\rho_1}{\|\rho_4\rho_1\|} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow$$

$$(1 - a_1^2)(x_{\rho_1} - x_{\rho_4})^2 - a_1^2(y_{\rho_1} - y_{\rho_4})^2 = 0 \tag{2.45}$$

$$(1 - a_2^2)(y_{\rho_1} - y_{\rho_4})^2 - a_2^2(x_{\rho_1} - x_{\rho_4})^2 = 0 \tag{2.46}$$

Solving equations (2.45) and (2.46) for x_{ρ_4} we get:

$$\begin{aligned} (1 - a_2^2)(1 - a_2^2)(x_{\rho_1} - x_{\rho_4})^2 - a_1^2 a_2^2 \\ (x_{\rho_1} - x_{\rho_4})^2 = 0 \\ \Rightarrow x_{\rho_4} = x_{\rho_1} \end{aligned} \tag{2.47}$$

Substituting (2.47) in equation (2.46), we get:

$$(1-a_2^2)(y_{\rho_1}-y_{\rho_4})^2 - a_2^2(x_{\rho_1}-x_{\rho_1})^2 = 0$$
$$\Rightarrow y_{\rho_4} = y_{\rho_1} \tag{2.48}$$

□

Corollary 3. *ρ_4 is within the convex hull of the virtual goals ρ_1, ρ_2, and ρ_3 if $\angle\rho_2\rho_1\rho_3 < 120°$.*

Proof. In Figure 2.7, let $\lambda = \angle\rho_2\rho_1\rho_3 < 120°$. This yields to (Boltyanski et. al, 1999, p. 236):

$$\angle\rho_1\rho_4\rho_3 = \angle\rho_1\rho_4\rho_2 = \angle\rho_2\rho_4\rho_3 = 120° \tag{2.49}$$

Hence, ρ_4 is within the convex hull of ρ_1, ρ_2, and ρ_3. □

The location of ρ_4 with regards to the location information of ρ_1 and ρ_3[7] is computed as:

$$\rho_4 = \begin{bmatrix} x_{\rho_1} \\ y_{\rho_1} - \|\rho_3\rho_1\| \times cos(\frac{\lambda}{2}) + \frac{(\|\rho_3\rho_1\| \times sin(\frac{\lambda}{2}) \times cos(\alpha))}{sin(\alpha)} \end{bmatrix} \tag{2.50}$$

2.3.1 Transformation of $\mathscr{VG}$ elements

Section 2.3 demonstrates a procedure to calculate a set of virtual goals based on their final desirable configuration. However, this procedure does not reflect the effect of the rotation of the final configuration on the location information of these virtual goals. We introduce this effect of the rotation of the final configuration to the location information of the virtual goals through the application of the transformational matrix:

$$R_z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.51}$$

where θ represents the angle of the rotation of the configuration.

We use equation (2.39) and equation (2.40) along with the transformational matrix in

[7] $\triangle\rho_2\rho_1\rho_3$ is an isosceles triangle, hence: $\|\rho_3\rho_1\| = \|\rho_2\rho_1\|$.

equation (2.51) to update the location information of ρ_2 and ρ_3 with regards to ρ_1. This yields to:

$$\begin{pmatrix} cos(\theta) \times (x_{\rho_1} - \|\rho_3\rho_1\| \times sin(\frac{\lambda}{2})) + sin(\theta) \times (y_{\rho_1} - \|\rho_3\rho_1\| \times cos(\frac{\lambda}{2})) \\ cos(\theta) \times (y_{\rho_1} - \|\rho_3\rho_1\| \times cos(\frac{\lambda}{2})) - sin(\theta) \times (x_{\rho_1} - \|\rho_3\rho_1\| \times sin(\frac{\lambda}{2})) \\ 1 \end{pmatrix} \tag{2.52}$$

Similarly, we use equation (2.50) and equation (2.51) to update the location information of ρ_4 to:

$$\begin{pmatrix} sin(\theta) * (y_{\rho_1} - (\|\rho_3\rho_1\| \times cos(\frac{\lambda}{2})) + \frac{(\|\rho_3\rho_1\| \times sin(\frac{\lambda}{2}) \times cos(\alpha))}{sin(\alpha)}) + x_{\rho_1} \times cos(\theta) \\ cos(\theta) * (y_{\rho_1} - (\|\rho_3\rho_1\| \times cos(\frac{\lambda}{2})) + \frac{(\|\rho_3\rho_1\| \times sin(\frac{\lambda}{2}) \times cos(\alpha))}{sin(\alpha)}) - x_{\rho_1} \times sin(\theta) \\ 1 \end{pmatrix} \tag{2.53}$$

2.4 Discussion

In this chapter, we presented a systematic approach to the decomposition of a mission of a multi-robot system using the distributions of the robotic agents and their task space. The decomposition of a mission expedites the decision-making process of individual agents. More specifically, it reduces the cardinality of the task space to a number of virtual goals that equates the total number of robotic agents. Therefore, robots need less time to make their decisions. It also has a substantial influence on the coordination of the agents. For instance, a rescue mission with a considerably large number of subtasks (e.g., lifeboats or trapped miners) can be decomposed into a number of regions. This results in a decision-making and coordination process that utilizes the location information of these regions to allocate subtasks to the robotic agents.

An important aspect of the decomposition process is its structural flexibility where certain refinement stages are bypassed based on the specification of a mission. As shown in Figure 2.1, steps such as subgrouping, linear, and isogonic decompositions are highly mission specific. Once the type of a mission is realized, the next step in the process determines whether the subgrouping of the task space is necessary. If positive, this step is executed otherwise it is bypassed. The outcome of this step is further processed to verify if representatives of subgroups are sufficient for the decision-making of the robotic agents (see Chapter 3) and the coordination of the robotic team (see Chapter 4). The conclusion of the

verification determines any additional step that is required for the generation of the virtual goals.

Chapter 3

Robotic Agent Decision Engine

In Chapter 2, we demonstrated the decomposition steps to generate a set of virtual goals using the location information of the robotic agents and the subtasks that form the body of a mission. We explained that the set of virtual goals $\mathscr{VG}$ constitutes the common knowledge of the robotic team. More specifically, all robotic agents utilize the same set of virtual goals to cast their votes. The voting of the individual agents is an incremental process. It is analogous to the estimation of the contributions of the individual robots to the overall mission at every decision cycle. It is through this process of voting that the ranking of the agents and the subsequent decision on their assignments at every decision cycle is determined.

Decision engine is the backbone of a multi-robot system. Its proper design and implementation is paramount to the successful accomplishment of a mission. In particular, this component enables the system to exploit the available information to infer the further steps such as the group-consensus and the coordination of agents.

We assert that there are two exclusive states associated with an agent that is situated in the world and is engaged in a mission: the *external state* and the *internal state*. The external state relates an agent to the mission and the surrounding environment. In contrast, the internal state relates the agent to itself. This approach to the formulation of the decision engine empowers an agent to realize its spatial relation to a mission. Furthermore, it enables the agent to determine its capability to fulfill the delegated tasks.

Incorporation of the state of an agent into the decision mechanism has been subject to a rigorous research. The probabilistic approaches (see Thrun et al., 2001; Charniak, 1991) focus mainly on the external state and discard the role of the internal state (see Borenstein et al. (1996) for the review of the topic). The requirement of *a priori* information in

the form of probability density functions (e.g., uniform probability distribution of success to every region of the environment at the commencement of a mission) is another subtlety of these approaches (e.g., Charnial, 1991; Furukawa et al., 2006; Chung and Furukawa, 2009). However, availability of *a priori* information is questionable if the environment and the task space are highly dynamic. Furthermore, the precision of the prediction is highly dependent on the size of *a priori* information and the result of the decision is misleading if this information is small.

The game theoretic frameworks attempt to model priors through a set of predetermined payoff values (e.g., Amigoni and Troiani, 2010). In other words, they transform the inherent dynamic of the decision-making problem into a highly deterministic formulation. They also have the shortcoming of high computational complexity due to the requirement of an exhaustive search of all possible outcomes of the game.

In the reminder of this chapter we formulate a decision mechanism that utilizes the internal and the external states of the agents to estimate their contributions at the individual-level. The overall formulation of the decision engine is presented in section 3.1. We alleviate the requirement of *a priori* information through incorporation of an opportunistic ranking module in the external state component of the decision engine in section 3.2. Section 3.3 presents the formal analysis of the proposed decision engine. We conclude this chapter in section 3.4.

3.1 Formal Representation

Although there is a correspondence between the results of the estimates of the internal and the external states of an agent, these results are conditionally independent. In particular, the result of the estimate of a state does not imply the computational outcome of the other state. For instance, the estimate of the external state to allocate a task to an agent may contradict the estimate of the internal state of the agent for the same task. This contradiction is plausible if the available energy to an agent is insufficient to navigate the path to the designated task. It is also possible for an agent to miss a specific equipment (e.g., an end-effector) to perform a task. Hence, it is crucial for a decision mechanism to explicitly recognize the independence of the results of the estimates of the internal and the external states during the decision process. This results in incompatibility of the additive incorporation of the internal and the external states. More specifically, the additive incorporation of

the estimates of these states provides the agent with an inaccurate and a misleading result. We further elaborate the conditional independence of the states of an agent through the following example.

Let assume the external state of the robotic agent r_i estimate a 100% of success if r_i is assigned with the virtual goal $\rho_j \in \mathscr{VG}$. However, the internal state of r_i realizes the lack of sufficient energy to reach ρ_j and hence rank this virtual goal with 0% estimate of success. It is apparent that the additive incorporation of the internal and the external states of r_i provides the agent with a misleading estimate to rank the virtual goal ρ_j with a 100% of success.

We address this incompatibility through the multiplicative incorporation of the internal and the external states into the decision mechanism:

$$\pi_i(\rho_j) = \frac{1}{\eta} \prod \psi_i(r_i, \rho_j)\phi_i(r_i, \rho_j),\ \forall \rho_j \in \mathscr{VG} \tag{3.1}$$

where
r_i: i^{th} robotic agent.
$\rho_j \in \mathscr{VG}$: j^{th} element of the set of virtual goals $\mathscr{VG}$.
$\pi_i(\rho_j)$: vote of the i^{th} agent for the virtual goal $\rho_j \in \mathscr{VG}$.
$\psi_i(r_i, \rho_j)$: the external state component (see section 3.2).
$\phi_i(r_i, \rho_j)$: the internal state component (see Appendix C).
Normalization factor $\frac{1}{\eta}$ ensures that the votes of an agent for the available virtual goals sum to 1:

$$\sum_{\rho_j \in \mathscr{VG}} \pi_i(\rho_j) = 1,\ \forall \pi_i(\rho_j) \geq 0,\ i = 1 \ldots n \tag{3.2}$$

where n represents the total number of robots.

Robotic agents use the decision engine in equation (3.1) to independently rank the virtual goal $\rho_j \in \mathscr{VG}$ incrementally and at every decision cycle. As a result, every individual agent maintains a number of vote values that is directly proportional to the cardinality of the set of virtual goals $|\mathscr{VG}|$. These values form the vote profile Π_i of a robotic agent.

Definition 5 (Vote Profile)**.** *Vote profile* Π_i *of the* i^{th} *robotic agent comprises the votes of*

the agent for the virtual goal $\rho_j \in \mathscr{VG}$*:*

$$\Pi_i = \{\pi_i(\rho_j) : \rho_j \in \mathscr{VG},\ \pi_i(\rho_j) \geq 0\} \tag{3.3}$$

3.2 External State Component

The external state component $\psi_i(r_i, \rho_j)$ in equation (3.1) ranks the contribution of an agent to a mission with respect to a given virtual goal using the location information of the robot and the virtual goal. It consists of the *default* and the *opportunistic* ranking modules.

Definition 6 (Default ranking module)**.** *Given a set of virtual goals* $\mathscr{VG}$ *at a decision cycle* $\underline{t}$*, the default ranking represents the estimate of the success of the agent to participate in the mission using the virtual goal* $\rho_j \in \mathscr{VG}$ *at decision cycle* $\underline{t}$*.*

The default ranking module of the i^{th} agent $\pi_i^t(r_i \mapsto \rho_j)$ calculates the vote for a given virtual goal $\rho_j \in \mathscr{VG}$ based on the current distance of the agent to the virtual goal $d_i^{cur}(r_i, \rho_j)$ and the desired agent-to-virtual goal distance $d_i^{desired}(r_i, \rho_j)$ at decision cycle t:

$$\pi_i^t(r_i \mapsto \rho_j) = \begin{cases} 1 & d_i^{cur}(r_i, \rho_j) \leq d_i^{desired}(r_i, \rho_j) \\ \frac{d_i^{desired}(r_i, \rho_j)}{d_i^{cur}(r_i, \rho_j)} & Otherwise \end{cases} \tag{3.4}$$

The desired distance $d_i^{desired}(r_i, \rho_j)$ is a predetermined, fixed integer that corresponds to the interval:

$$0 < d_i^{desired}(r_i, \rho_j) \leq \lfloor d_i^{cur}(r_i, \rho_j) - (d_i^{cur}(r_i, \rho_j) - \lfloor d_i^{cur}(r_i, \rho_j) \rfloor) \rfloor \tag{3.5}$$

Equation (3.5) indicates that the desired distance $d_i^{desired}(r_i, \rho_j)$ is bounded with an upper limit that is equal to the floor of the integer portion of the current distance of the robotic agent to the virtual goal. Furthermore, equation (3.5) expresses that zero is not a permissible choice for the value of the desired distance. This is due to the fact that a desired distance that is set to zero yields the result of the computation of equation (3.4) to be steadily zero if $d_i^{cur}(r_i, \rho_j) > d_i^{desired}(r_i, \rho_j)$.

The choice of the value of the desired distance is highly domain-specific. This value is

influenced by the type of information that is represented by the virtual goals. For example, the value of the desired distance is as close as possible to zero (e.g., it is set to one) when a virtual goal marks a specific location for an agent to attend. On the other hand, an approximation of the vicinity of the agent to the virtual goal suffices if the virtual goal represents a region.

The superscript t in equation (3.4) refers to the decision cycle t. This implies that agent r_i considers the distance to a virtual goal ρ_j that is encountered at time t. This distance is calculated based on the location information of the robotic agent and the virtual goal. The current distance of an agent to a virtual goal is dependent on its ability to detect obstacles. In particular, this distance is unaffected by the presence of an obstacle that is beyond the detection range of an agent. Therefore, the current distance is calculated as:

$$d_i^{cur}(r_i, \rho_j) = \begin{cases} w_i \|\rho_j - r_i\| & No\ obstacle \\ \\ w_i \|p - r_i\| + w_i \|\rho_j - p\| & Otherwise \end{cases} \tag{3.6}$$

where p represents the next temporary location that r_i selects reactively to avoid collision with an obstacle.

Definition 7 (Opportunistic ranking module)**.** *Given a set of virtual goals $\mathscr{VG}$ at a decision cycle $\underline{t}$, the opportunistic ranking represents the vote values of the agent $\forall \rho_j \in \mathscr{VG}$ that are calculated at $\underline{t-1}$.*

The opportunistic ranking module $\omega_i(\rho_j)$ incorporates the estimate of the success of an agent for a given virtual goal $\rho_j \in \mathscr{VG}$ that is acquired in previous decision cycle:

$$\omega_i(\rho_j) = C + \pi_i^{t-1}(\rho_j) \tag{3.7}$$

where $C \in [0 \ldots 1]$ is a constant that initializes the opportunistic ranking module of the external state component of the decision engine at the commencement of a mission. This initialization value is necessary to avoid an unexpected behavior of the decision mechanism in the first decision cycle due to an undefined value of the opportunistic ranking module. There is no restriction on the choice of this initialization value from the given interval. However, values larger than zero suggest the prioritization of the virtual goals. Therefore, we intentionally use the value zero to prevent any contingent assumption of availability of

a priori information. As a result, the confidence of the robotic agents is constructed based solely on the evolution of their votes. This evolution reflects the result of the independent decisions of the agents at a given decision cycle..

We use euqation (3.4) and equation (3.7) to express the external state component as:

$$\psi_i(r_i, \rho_j) = \pi_i^t(r_i \mapsto \rho_j) + \omega_i(\rho_j) \tag{3.8}$$

$$= \pi_i^t(r_i \mapsto \rho_j) + C + \pi_i^{t-1}(\rho_j),\ \forall \rho_j \in \mathscr{VG} \tag{3.9}$$

The conceptual diagram of the external state component of the decision engine is presented in Figure 3.1. It calculates the default ranking of an agent in conjunction with a set of virtual goals $\mathscr{VG}$. Next, it obtains the final vote of a virtual goal $\rho_j \in \mathscr{VG}$ through the cumulative sum of its default and the opportunistic rankings. In addition, it updates the opportunistic ranking of the agent using its final votes at a given decision cycle. More specifically, this component overwrites the opportunistic rankings that are acquired in the previous decision cycle with the votes in present decision cycle. As a result, it enhances the decision engine to an evolving mechanism to determine the confidence of the agents on a set of virtual goals as the state of a mission progresses.

3.2.1 Effect of the Opportunistic Ranking Module

Let Table 3.1 represent the vote values of the robotic agent r_1 for a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ at the first decision cycle, $t = 1$. This implies that the opportunistic

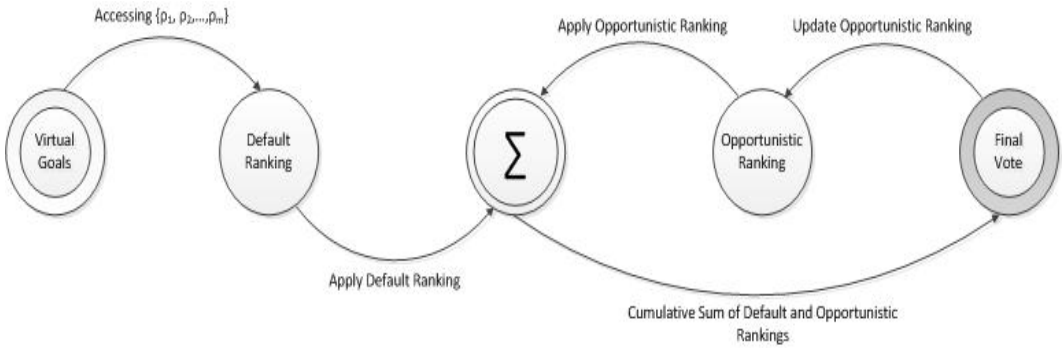

Figure 3.1: The conceptual diagram of the external state component $\psi_i(r_i, \rho_j)$.

Table 3.1: The estimate of the external state component of agent r_1 for a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ at decision cycle $t = 1$

r_1	ρ_1	ρ_2	ρ_3
$\pi_1^t(r_1 \mapsto \rho_j)$	0.41	0.16	0.43
$\omega_1(\rho_j)$	0.00	0.00	0.00
$\pi_1(\rho_j)$	0.41	0.16	0.43

ranking module $\omega_1(\rho_j) = 0,\ \forall \rho_j \in \mathscr{VG}$ (see equation 3.7).[1] Therefore, entries of Table 3.1 represent the vote profile of r_1 that is solely calculated using its default ranking module (see equation 3.4 through equation 3.6). These vote values are calculated using the location information of r_1 and $\rho_j \in \mathscr{VG}$. The entries of Table 3.1 show that the decision engine of r_1 ranks the virtual goal ρ_3 with the highest estimate of success since $\pi_1(\rho_3) = 0.43 > \pi_1(\rho_1) = 0.41 > \pi_1(\rho_2) = 0.16$.[2]

Let Table 3.2 represent the vote profile of r_1 calculated using the default ranking module of the agent in the next decision cycle $t + 1$. The entries of Table 3.2 indicate that the decision mechanism of r_1 ranks ρ_1 with highest estimate in this decision cycle. However, a comparison of the entries of Table 3.1 and Table 3.2 reveals that the modification of the estimate of the success of r_1 is due to a slight change of the votes between these two consecutive decision cycles.

The opportunistic ranking module enables the decision mechanism to reduce the possibility of an unnecessary modification of the ranking of the virtual goals. In particular, this module represents the incremental evolution of the confidence of the robotic agents on available virtual goals (see Definition 7 and equation 3.7). The last row entry of Table 3.1 corresponds to the vote values of r_1 at decision cycle $t = 1$. These values constitute the opportunistic ranking of the virtual goals in decision cycle $t + 1$ (see Figure 3.1). The

Table 3.2: The estimate of the default ranking module $\pi_1^{t+1}(r_1 \mapsto \rho_j)$ of r_1 for a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ at decision cycle $t + 1$

r_1	ρ_1	ρ_2	ρ_3
$\pi_1^{t+1}(r_1 \mapsto \rho_j)$	0.43	0.145	0.425

[1] This explains the requirement of the initialization value C in equation 3.7.

[2] $\pi_i(\rho_j)$ represents the vote of i^{th} agent for the j^{th} virtual goal (see equation 3.1).

decision engine of r_1 utilizes these values along with the default ranking of the virtual goals at decision cycle $t+1$ (i.e., the entries of Table 3.2) to calculate the vote profile of r_1 using equation (3.9):

$$\pi_1(\rho_1) = \pi_1^{t+1}(r_1 \mapsto \rho_1) + \omega_1(\rho_1) = 0.840$$
$$\pi_1(\rho_2) = \pi_1^{t+1}(r_1 \mapsto \rho_2) + \omega_1(\rho_2) = 0.305$$
$$\pi_1(\rho_3) = \pi_1^{t+1}(r_1 \mapsto \rho_3) + \omega_1(\rho_3) = 0.855$$

These votes are normalized to obtain:

$$\pi_1(\rho_1) = \pi_1^{t+1}(r_1 \mapsto \rho_1) + \omega_1(\rho_1) = 0.510$$
$$\pi_1(\rho_2) = \pi_1^{t+1}(r_1 \mapsto \rho_2) + \omega_1(\rho_2) = -0.025$$
$$\pi_1(\rho_3) = \pi_1^{t+1}(r_1 \mapsto \rho_3) + \omega_1(\rho_3) = 0.525$$

This normalization procedure continues until the votes comply with equation (3.2). This yields the values:

$$\pi_1(\rho_1) = \pi_1^{t+1}(r_1 \mapsto \rho_1) + \omega_1(\rho_1) = 0.4925$$
$$\pi_1(\rho_2) = \pi_1^{t+1}(r_1 \mapsto \rho_2) + \omega_1(\rho_2) = 0.0000$$
$$\pi_1(\rho_3) = \pi_1^{t+1}(r_1 \mapsto \rho_3) + \omega_1(\rho_3) = 0.5075$$

Table 3.3 shows the vote profile of r_1 at decision cycle $t+1$. The entries of Table 3.3 indicate that the decision engine of r_1 ranks ρ_3 with highest vote at decision cycle $t+1$ as well.

Table 3.3: The estimate of the external state component of agent r_1 after the incorporation of the opportunistic ranking module $\omega_i(\rho_j)$.

r_1	ρ_1	ρ_2	ρ_3
$\pi_1^{t+1}(r_1 \mapsto \rho_j)$	0.4300	0.145	0.4250
$\omega_1^t(\rho_j \mid r_1)$	0.4100	0.160	0.4300
$\pi_1(\rho_j)$	0.5100	-0.025	0.5250
$\pi_1(\rho_j)$	0.4925	0.000	0.5075

3.3 Formal Analysis

In this section, we formally analyze the performance of the decision engine of the robotic agents. In particular, we demonstrate the capability of this mechanism to infer the best choice of the virtual goal of the agents at the individual-level. We define the best choice of the virtual goal of an agent as follows.

Definition 8 (Agent Best Choice)**.** *Given a set of virtual goals $\mathscr{VG} = \{\rho_1 \dots \rho_m\}$, and the vote profile Π_i that comprises the vote values $\pi_i(\rho_j)$ of the i^{th} robotic agent, the best choice of virtual goal $\hat{\rho} \in \mathscr{VG}$ satisfies:*

$$\pi_i(\hat{\rho}) \geq \pi_i(\rho_j),\ \forall \rho_j \in \mathscr{VG},\ \hat{\rho} \in \mathscr{VG} \tag{3.10}$$

This implies that:

$$\hat{\rho} \in argmax\ \pi_i(\rho_j),\ \forall \rho_j \in \mathscr{VG},\ \hat{\rho} \in \mathscr{VG} \tag{3.11}$$

Theorem 4 (Optimal Choice)**.** *The vote profile Π_i consists of at least one vote value that corresponds to the best choice of virtual goal of the i^{th} agent at every decision cycle.*

Proof. Let $\mathscr{VG}$ represent a set of virtual goals. Then $\forall \rho_j \in \mathscr{VG}$, $\pi_i(\rho_j)$ is the vote of i^{th} robotic agent that is calculated using equation (3.1). Let $\Pi_i = \{\pi_i(\rho_j) : \rho_j \in \mathscr{VG}\ ,\ \pi_i(\rho_j) \geq 0\}$ represent the vote profile of the i^{th} robotic agent. If Π_i is a singleton set, then the proof is trivial. However, if $|\Pi_i| > 1$ where $|\Pi_i|$ denotes the cardinality of Π_i, there exists at least one element $\pi_i{}'(\rho_j) \in \Pi_i$ such that:

1. $\pi_i{}'(\rho_j) > \pi_i(\rho_j),\ \forall \pi_i(\rho_j) \in \Pi_i$, then $\pi_i{}'(\rho_j)$ is the vote value that strongly dominates all the elements of Π_i. Hence it is the best choice of the agent.

2. $\pi_i{}'(\rho_j) \geq \pi_i(\rho_j),\ \forall \pi_i(\rho_j) \in \Pi_i$, then $\pi_i'(\rho_j)$ weakly dominates all the elements of Π_i. This vote satisfies equation (3.10) and hence it is the best choice of the agent.

3. $\exists \pi_i(\rho_j) \in \Pi_i,\ \pi_i{}'(\rho_j) = \pi_i(\rho_j)$, then the i^{th} agent is indifferent to the two vote values and either choice is the best voted virtual goal.

□

Theorem 1 and Corollary 1 in section 2.2.3 demonstrate that the linear decomposition enables a multi-robot system to generate a set of virtual goals to optimize the relocations of the robotic agents. However, we assume a special case of an obstacle-free environment to prove these optimal relocations of the agents. We generalize these results through the following theorem. More specifically, we demonstrate that the results of Theorem 1 and Corollary 1 is unaffected by the presence of obstacles if the robotic agents are capable of determining their best choices of virtual goals (see section 3.3, Theorem 4).

Theorem 5 (Generalization on the Presence of the Obstacles)**.** *In an environment that comprises obstacles of arbitrary numbers and shapes, the weighted sum of Euclidean distances of the robotic agents to their virtual goals $\rho_j \in \mathscr{VG}$ is minimized if the agents are able to elect their respective best choices of virtual goals.*

Proof. We consider two cases:

1. *All robots attend their corresponding virtual goals without avoiding collision with obstacles*:
   This implies that the best choice of the virtual goal of every robotic agent is the virtual goal that is generated using its location information. Therefore:[3]

$$min \sum_{i=1}^{n} w_i \|\rho_i - r_i\|,\ \rho_i \in \mathscr{VG} \tag{3.12}$$

2. *Some of the robotic agents, $r_j,\ j = 1 \ldots m,\ m \leq n$ need to avoid collision with obstacles, where n represents the total number of robots*:
   Equation (3.12) holds for the agents that reach their corresponding virtual goals without avoiding collision. Hence:

$$min \sum_{i=1}^{n-m} w_i \|\rho_i - r_i\|,\ \rho_i \in \mathscr{VG} \tag{3.13}$$

   However, for the robotic agents $r_j,\ j = 1 \ldots m,\ m \leq n$ it is either the case that their corresponding virtual goals are still the virtual goals that result in shortest travel

[3]This situation is analogous to an obstacle-free environment since robots follow their paths to their virtual goals without performing any collision avoidance. As a result, Theorem 1 and Corollary 1 hold.

distances after avoiding collision with obstacles:

$$min \sum_{j=1}^{m} w_j \|\rho_j - r_j\|,\ \rho_j \in \mathscr{VG} \tag{3.14}$$

which yields to:

$$\begin{aligned} min[\sum_{i=1}^{n-m} w_i \|\rho_i - r_i\| \ + \sum_{j=1}^{m} w_j \|\rho_j - r_j\|] \\ \Rightarrow min \sum_{i=1}^{n} w_i \|\rho_i - r_i\| \ \ \rho_i,\ \rho_j \in \mathscr{VG} \end{aligned} \tag{3.15}$$

or it is the case that some of the robotic agents $r_j,\ j = 1 \ldots k,\ k \leq m$ elect alternative virtual goals $\hat{\rho_j} \in \mathscr{VG}$ as their best choices of the virtual goals after avoiding collision with obstacles:

$$\sum_{j=1}^{k} w_j \|\hat{\rho_j} - r_j\| \ \leq \sum_{j=1}^{k} w_j \|\rho_j - r_j\|,\ \hat{\rho_j},\ \rho_j \in \mathscr{VG} \tag{3.16}$$

This results in:

$$min[\sum_{i=1}^{m-k} w_i \|\rho_i - r_i\| \ + \sum_{j=1}^{k} w_j \|\hat{\rho_j} - r_j\|],\ \rho_i,\ \hat{\rho_j} \in \mathscr{VG} \tag{3.17}$$

Using equation (3.13) and equation (3.17), we get:

$$min \sum_{i=1}^{n-m} w_i \|\rho_i - r_i\| + min[\sum_{i=1}^{m-k} w_i \|\rho_i - r_i\| \ + \sum_{j=1}^{k} w_j \|\hat{\rho_j} - r_j\|] \tag{3.18}$$

$$\Rightarrow min \sum_{j=1}^{n} w_j \|\rho_j - r_j\|,\ \rho_j,\ \hat{\rho_j} \in \mathscr{VG} \tag{3.19}$$

□

Lemma 6 (Agent Best Choice Complexity)**.** *It takes no longer than $O(m)$ for an agent to find its best choice of virtual goal.*

Proof. Let $|\Pi_i| = m$ represent the cardinality of the vote profile of i^{th} robotic agent. It is

apparent that to find the best choice of the virtual goal is equivalent to ascertaining the highest voted virtual goal $\rho_j \in \mathscr{VG}$ in Π_i. This is done in a linear time. Hence $O(m)$. □

Section 3.3, Theorem 4 underlines a situation where the robotic agents have equal confidence on multiple virtual goals. This condition may suggest the necessity for further analysis of the vote profile before inferring the best choice of virtual goal of an agent. However, the following theorem proves that such an analysis of the solution space is not required since any other possible highly voted choices of the virtual goals is at most as good as the virtual goal that is determined as the best choice of virtual goal of the agent.

Theorem 6. For a robot **r** with its best choice of virtual goal denoted by $\hat{\rho} \in \mathscr{VG},\ |\mathscr{VG}| \geq 1$, given any $\rho^* \in \mathscr{VG},\ \pi_r(\rho^*) \geq 0$, it is true that: $\pi_r(\rho^*) \leq \pi_r(\hat{\rho}),\ \forall \rho^* \in \mathscr{VG}$.

Proof. If $\mathscr{VG}$ is a singleton set, then the proof is trivial. Let $|\mathscr{VG}| > 1,\ \hat{\rho} \in \mathscr{VG}$. If $\exists \rho^* \in \mathscr{VG}$ with the vote value higher than that of the $\hat{\rho}$, using equation 3.1 we have:

$$\begin{aligned}
\pi_r(\hat{\rho}) &= argmax_{\rho \in \mathscr{VG}}(\frac{1}{\eta}\prod \psi(r,\rho)\phi(r,\rho)) \\
&= argmax_{\rho \neq \rho^*}[\frac{1}{\eta}\prod \psi(r,\rho)\phi(r,\rho)] \\
&\quad \times [\frac{1}{\eta}\prod \psi(r,\rho^*)\phi(r,\rho^*)] \\
&= argmax_{\rho \neq \rho^*}[(\pi_r^t(r \mapsto \rho) + \omega_r(\rho)) \times \pi_r(e_r \mapsto \rho)] \\
&\quad \times [(\pi_r^t(r \mapsto \rho^*) + \omega_r(\rho^*)) \times \pi_r(e_r \mapsto \rho^*)] \\
&< argmax_{\rho \neq \rho^*}[(\pi_r^t(r \mapsto \rho^*) + \omega_r(\rho)) \times \pi_r(e_r \mapsto \rho^*)] \\
&\quad \times [(\pi_r^t(r \mapsto \rho^*) + \omega_r(\rho^*)) \times \pi_r(e_r \mapsto \rho^*)] \\
&= \pi_r(\rho^*)
\end{aligned} \tag{3.20}$$

A contradiction to the original assumption that $\hat{\rho}$ is the best choice of the virtual goal of the agent. □

3.4 Discussion

In this chapter, we presented our approach to formulation of the decision engine of the robotic agents. We showed how the state of an agent with respect to a delegated mission is viewed as a combination of two exclusive internal and external states. Furthermore, we

asserted the logical reasoning behind this mutual exclusiveness through the elaboration of the irrelevance of their additive incorporation.

We presented how the requirement of *a priori* information is prevented via the application of a simple opportunistic ranking module in the external state component of the decision engine. Specifically, we showed that the opportunistic module provides an autonomous independent agent with a reliable mechanism to estimate the incremental evolution of its confidence at every decision cycle. Furthermore, we showed that the computational complexity of the decision mechanism is linear to the cardinality of a set of virtual goals. We demonstrated that the mechanism is capable of ascertaining the best choice of virtual goal of the agents at the individual-level.

We focused primarily on the formulation of a decision mechanism that imposes the least travel distance on the agents to attend the virtual goals. The minimization of the travel distance has a significant influence on the performance of the agents. However, the votes that are estimated to minimize solely the distances do not thoroughly reflect the entire dynamics of a multi-robot system and the complexity of their task space. An alternative approach to the formulation of the external state component of the decision engine is an objective function that minimizes the total travel time or the cost of the trade-off of the energy and the time to accomplish a task. This modification of the decision engine requires the inclusion of several other factors to calculate the votes. These factors vary from the velocity bound imposed on an agent, to the time constraints and ordering/priorities among the subtasks of the overall task space.

Chapter 4

Group Coordination

In Chapter 2, we demonstrated the decomposition process of a mission into a set of virtual goals to facilitate the distribution of the overall task space among autonomous robotic agents. Chapter 3 advanced on utilization of this set to enable a multi-robot system to employ individual members of the team, distributively, as autonomous independent decision-making units. Furthermore, we demonstrated that the vote profiles of these agents consist of their best choices of the virtual goals (see Chapter 3, Definition 5, and Theorem 4 through Theorem 6).

However, the best choices of virtual goals are determined at the individual-level and do not consider the allocation strategy that suites the entire team. Assignments of two or more robots to the same task or any constraint on the number of attendees of a specific virtual goal are the examples where a multi-robot system requires coordinating these independent decisions. Furthermore, the distribution of the robots or the virtual goals can be biased to a particular region of the field of the operation. As a result, the virtual goals or the agents that are in relatively farther distances from the high density region are discarded. This influences the decision engine of the robotic agents to rank these outliers unfavorably. Therefore, it is crucial for the system to bring the independent decisions of the individual robots together to infer the coordination strategy at the group-level in every decision cycle.

Definition 9 (Profile Matrix)**.** *Profile matrix* $\Phi_{n\times m}$ *of a group of robotic agents* $r_i,\ i = 1 \dots n$*, is a matrix where every row entry corresponds to the vote profile of an individual*

agent r_i:

$$\Phi_{n\times m} = \begin{bmatrix} \Pi_1 \\ \Pi_2 \\ \vdots \\ \Pi_n \end{bmatrix} = \begin{bmatrix} \pi_1(\rho_1) & \pi_1(\rho_2) & \cdots & \pi_1(\rho_m) \\ \pi_2(\rho_1) & \pi_2(\rho_2) & \cdots & \pi_2(\rho_m) \\ \vdots & & \ddots & \vdots \\ \pi_n(\rho_1) & \pi_n(\rho_2) & \cdots & \pi_n(\rho_m) \end{bmatrix} \tag{4.1}$$

where n and m represent the number of robotic agents and the cardinality of a set of virtual goals, $|\mathscr{VG}|$.

Furthermore, the number of robots that are permitted to attend a specific virtual goal is an important factor to determine the underlying process of the coordination of an allocation strategy.

Definition 10 (Virtual Goals Constraint q)**.** *Given a team of n robotic agents and a set of virtual goals $\mathscr{VG}$, $|\mathscr{VG}| = m$, constraint $0 \leq q \leq n$ imposed on $\rho_j \in \mathscr{VG}$, $j = 1 \dots m$, specifies the permissible number of robots assigned to ρ_j at a given decision cycle.*

We use the profile matrix $\Phi_{n\times m}$ and the constraint q to define the optimal coordination strategy as follows.

Definition 11 (Optimal Coordination Strategy)**.** *Given a group of n robotic agents r_i , $i = 1 \dots n$, a set of virtual goals $\mathscr{VG} = \{\rho_1, \dots, \rho_m\}$, the profile matrix $\Phi_{n\times m}$, and the constraint q, an optimal coordination strategy distributes virtual goals $\rho_j \in \mathscr{VG}, j = 1 \dots m$, among agents such that:*

$$\upsilon \in argmax \sum_{i=1}^{n} \sum_{j=1}^{m} \Phi_{ij} \ : \ \forall \rho_k, \ \rho_j \in \upsilon, \tag{4.2}$$

$$\rho_k = \rho_j \rightarrow |\upsilon \setminus \upsilon - \{\rho_k\}| \leq q, \ k, \ j \leq m \tag{4.3}$$

In equation (4.2), Φ_{ij} refers to a specific entry of the profile matrix (see Chapter 4, Definition 9). Moreover, υ denotes a subset of $\rho_j \in \mathscr{VG}$ where the cumulative sum of the votes of robotic agents is maximum. Equation (4.3) ensures that the number of attendees of every virtual goal ρ_j does not exceed the constraint q (see Chapter 4, Definition 10). Furthermore, υ is a $1 \times n$ vector where every column entry corresponds to an individual agents.

There exists a number of approaches to the formulation of the coordination of the allocation strategy of a multi-robot system. The Kuhn-Munkres algorithm (see Kuhn, 1955) and the Hungarian algorithm (see Munkres, 1957, and Appendix D.2, Algorithm 7) model the task allocation strategy using a complete bipartite graph.[1] However, a complete bipartite graph requires that the size of the robotic team equals the cardinality of the task space. This limits the scope of these algorithms to the scenarios where an equal number of robots and tasks are introduced.

Market-based approaches (see section 1.3 for the review of these approaches) are widely used to coordinate the multi-robot systems. They are classified mainly under two dominant coordination categories. One category allocates tasks using a one time voting procedure that resembles an auction (e.g., Gerkey and Mataric, 2002; Kose et al., 2004). All the robotic agents vote for the entire task space once at the commencement of a mission. Therefore, the allocated tasks are fixed after the coordination procedure is finalized. The other category performs the coordination of the robotic agents using an extended allocation procedure. Multiple times voting (e.g., Tovey et al., 2005) and peer-to-peer trading (e.g., Rabideau et al., 1999) are the examples of this category. However, the market-based approaches require introducing tasks one at a time and finalize the allocation before moving to the next iteration of the allocation.

We address these issues through the formulation of two coordination strategies. These are *agents votes maximization* (see section 4.1) and *profile matrix permutations* (see section 4.2). Similar to the Hungarian algorithm, the agents votes maximization strategy achieves the time complexity of $O(n^3)$ when the number of robots equals the cardinality of task space.[2] In addition, this strategy is capable of addressing the scenarios where the number of robots and the cardinality of task space are not the same. It utilizes the profile matrix of a multi-robot system (see Chapter 4, Definition 9) to prevent the limitation of the complete bipartite graph representation. Although Bourgeois and Lassalle (1971) extend the Hungarian algorithm to address the latter issue, their algorithm does not distribute the tasks evenly among the agents. Furthermore, the agents votes maximization strategy addresses the situation where the available tasks are constrained with the maximum number of allocated agents (see section 4.1.2, Lemma 7 through Lemma 9).

[1] A full description of bipartite graphs and their applications is found in (Bondy and Murty, 2009).

[2] It is $O(n^4)$ in case of Kuhn-Munkres algorithm

In contrast, the profile matrix permutation strategy achieves an optimum allocation of tasks via calculation of the permutation of the votes of robotic agents where the cumulative sum of these votes is maximum.

4.1 Agents Votes Maximization Strategy

This strategy coordinates a multi-robot system through the redirection of agents with lower vote values to the virtual goals that are least favored with the robotic agents that achieve the highest votes in the system.

Algorithm 4 summarizes the agents votes maximization coordination strategy. The profile matrix and the constraint q are the inputs to this algorithm. It allocates the virtual goals (see Chapter 2, Definition 1) based on the highest votes of the individual robots. More specifically, agents with the higher vote values are treated with higher priority to assign the virtual goals at a given decision cycle. In addition, the algorithm tracks the number of agents that are assigned to a virtual goal ρ_j in the array VG. This ensures that the number of attendees of a virtual goal does not exceed the constraint q. Every time a robot is assigned to a virtual goal ρ_j the corresponding entry of this virtual goal $VG[j]$ is incremented. Subsequently, the vote of the robotic agent that is allocated to this virtual goal in an iteration is set to -1. This prevents the same agent to be reselected for the same virtual goal in consecutive iterations. Next, the entries of the VG are compared with q. As a result, the entry of a virtual goal ρ_j where $VG[j] = q$ is removed from the profile matrix $\Phi_{n\times m}$. Moreover, the agents votes maximization strategy increments the entry of the i^{th} robotic agent in the array $Tagged$ (i.e., $Tagged[i]$) when this agent is assigned to a virtual goal. This ensures that every agent is allocated with at least one virtual goal. These robotic agents are not considered for the further allocation of the available virtual goals until the next iteration. It is apparent that every agent is assigned with exactly one virtual goal if $q = 1$.

Algorithm 4: Agents votes maximization coordination strategy

```
Data: Φ_{n×m}, Profile Matrix.
Data: q, Constraint on number attendees per virtual goal.
Data: m, total number of virtual goals.
begin
    count ← 0;
    while count ≤ m do
        for i = 1 : n do
            Tagged[i] ← −1;
        max ← −1;
        for i = 1 : n do
            for j = 1 : m do
                if Φ[i][j] > max then
                    if Tagged[i] == i then
                        continue;
                    else
                        max ← Φ[i][j];
                        row ← i;
                        column ← j;
        if Tagged[row] != row then
            Assign ρ_column to the robot r_row;
            Tagged[row] ← row;
            VG[column] ← VG[column] + 1;
            if VG[column] == q then
                count ← count + 1;
                if q == 1 then
                    Φ[row][column] ← −1; // i.e. Remove the corresponding entry
                else
                    Remove ρ_column from Φ_{n×m};
```

4.1.1 Numerical Example

Let the profile matrix for a group of robotic agents $r_i,\ i = 1 \dots 3$, and the set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ be:

$$\Phi_{3\times 3} = \begin{bmatrix} \Pi_1 \\ \Pi_2 \\ \Pi_3 \end{bmatrix} = \begin{bmatrix} 043 & 0.17 & 0.40 \\ 0.10 & 0.35 & 0.55 \\ 0.75 & 0.15 & 0.10 \end{bmatrix} \tag{4.4}$$

Let the constraint on the number of attendees of these virtual goals be $q = 1$. This implies that every virtual goal is allocated to exactly one agent.[3] Let the value -1 represent an

[3]This setting of the constraint q is for the simplicity of the example and does not imply an explicit restriction of the algorithm.

agent r_i where the allocation of this agent to a virtual goal is completed. Hence, the entry of the profile matrix with -1 implies a virtual goal that is allocated.

In equation (4.4), the highest vote corresponds to $\pi_3(\rho_1) = 0.75$ (see equation 3.1). Therefore, the virtual goal ρ_1 is allocated to the agent r_3 (i.e., $r_3 \leftarrow \rho_1$). Subsequently, the entry of r_3 and ρ_1 is set to -1. This updates the entries of the profile matrix $\Phi_{3\times3}$ to:

$$\begin{bmatrix} -1 & 0.17 & 0.40 \\ -1 & 0.35 & 0.55 \\ 0.75 & -1 & -1 \end{bmatrix} \tag{4.5}$$

In the matrix 4.5, it is apparent that $r_2 \leftarrow \rho_3$ since $\pi_2(\rho_3) = 0.55$. Therefore:

$$\begin{bmatrix} -1 & 0.17 & -1 \\ -1 & -1 & 0.55 \\ 0.75 & -1 & -1 \end{bmatrix} \tag{4.6}$$

This results in $r_1 \leftarrow \rho_2$ since $q = 1$. Hence, the final allocation of the agents becomes:

$$\Phi_{3\times3} = \begin{bmatrix} \Pi_1 \\ \Pi_2 \\ \Pi_3 \end{bmatrix} = \begin{bmatrix} 043 & 0.17 & 0.40 \\ 0.10 & 0.35 & 0.55 \\ 0.75 & 0.15 & 0.10 \end{bmatrix} \tag{4.7}$$

4.1.2 Complexity Analysis

Let n and $|\mathscr{VG}| = m$ represent the number of robotic agents and the cardinality of a set of virtual goals, respectively. Let q be the constraint imposed on the number of attendees of these virtual goals (see Chapter 4, Definition 10). There are three cases to consider:[4]

1. $q = 0$: no constraint is imposed on the number of attendees of a virtual goal $\rho_j \in \mathscr{VG}$.[5]
2. $q \geq 2$: number of attendees of these virtual goals is permitted to be ≥ 2.
3. *q=1*: every virtual goal is assigned to exactly one robot.

[4] We assume that q is the same for all the virtual goals.

[5] This implies that the robotic agents are permitted to attend any available virtual goal.

Lemma 7 ($q = 0$)**.** *Given n number of robots and a set of virtual goals $\mathscr{VG}$ with cardinality m where $q = 0$, $\forall \rho_j \in \mathscr{VG}$, it takes no longer than $O(mn)$ to complete the allocation of these virtual goals using the agents votes maximization strategy.*

Proof. Every virtual goal is permitted to be allocated to an arbitrary number of robotic agents. Therefore, it suffices to ascertain the best choice of the virtual goal of every agent (see Chapter 3, Definition 8) in its corresponding entry in the profile matrix $\Phi_{n\times m}$ (see Chapter 4, Definition 9). This is done in a linear time $O(m)$ since there are m number of virtual goals. Having n number of robots, the allocation of m virtual goals is completed in $O(mn)$. □

Lemma 8 ($q \geq 2$)**.** *Given n number of robots and a set of virtual goals $\mathscr{VG}$ with cardinality m where $q \geq 2$, $\forall \rho_j \in \mathscr{VG}$, it takes no longer than $O(m^3 q)$ to complete the allocation of these virtual goals using the agents votes maximization strategy.*

Proof. It is apparent that this setting requires to satisfy:

$$n \leq mq \tag{4.8}$$

Assuming equation (4.8) is satisfied, there are two cases to be considered:

1. *Virtual goals are allocated to q number of attendees and robots are allowed to be assigned to arbitrary number of virtual goals:* Using result obtained in Lemma 7, mn steps are required to assign first q robots to their respective virtual goals. However, the number of virtual goals is reduced by one every time the number of attendees of a virtual goal equals q (see Algorithm 4). On the other hand, the number of robots remains the same. Hence:

$$\begin{aligned} &mn + (m-1)n + (m-2)n + \dots \\ &= \frac{m(m+1)}{2} n \leq m^2 n \Rightarrow O(m^2 n) \end{aligned} \tag{4.9}$$

2. *Virtual goals are allocated to q number of attendees and robots are restricted to one virtual goal assignment:* Using Lemma 7, mn steps are required to assign first q robots to their respective virtual goals. The number of agents is reduced by q at this point. Furthermore, the number of virtual goals is reduced by 1 every time the number of

attendees of a virtual goal is q. This results in $(m-1)(n-q)$ steps. Continuing in the same fashion and replacing n by mq (see equation 4.8) we get:

$$\begin{aligned} & mn + (m-1)(n-q) + (m-2)(n-2q) + \dots \\ &= m^2q + (m-1)(mq-q) + (m-2)(mq-2q) + \dots \\ &= m^2q + mq[(m-1) + (m-2) + (m-3) + \dots] - \\ & \quad q[(m-1) + 2 \times (m-2) + 3 \times (m-3) + \dots] \end{aligned} \tag{4.10}$$

Expanding on the first term, we get:

$$\begin{aligned} & mq[(m-1) + (m-2) + (m-3) + \dots] \\ &= mq \times \sum_{1}^{m-1} m = \frac{m^3q - m^2q}{2} \end{aligned} \tag{4.11}$$

Expanding on the second term, we get:

$$\begin{aligned} & -q[(m-1) + 2 \times (m-2) + 3 \times (m-3) \dots] \\ &= -q[\sum_{i=1}^{m-1} (m - (i+1)) \times i] \\ &= -q[\sum_{i=1}^{m-1} m \times i - \sum_{i=1}^{m-1} (i+1) \times i] \\ &= -q[m \sum_{i=1}^{m-1} i - (\sum_{i=1}^{m-1} i^2 + \sum_{i=1}^{m-1} i)] \\ &= -q[\frac{m^2(m-1)}{2} - [(\frac{m(m+1)(2m+1)}{6} - m^2) \\ & \quad + \frac{m^2 - m}{2}]] = -q(\frac{m^3 - 3m^2 + 2m}{6}) \end{aligned} \tag{4.12}$$

Replacing equations (4.11) and (4.12) in equation (4.10) yields to:

$$\begin{aligned} & m^2q + \frac{m^3q - m^2q}{2} - (\frac{m^3q - 3m^2q + 2mq}{6}) \\ &= \frac{1}{3}(m^3q - mq) \leq m^3q \Rightarrow O(m^3q) \end{aligned} \tag{4.13}$$

□

Lemma 9 (Special Case: $q = 1$)**.** *It takes no longer than $O(m^3)$ to complete a one-to-one allocation of the robots to the virtual goals.*

Proof. This setting requires to satisfy:

$$n = |\mathscr{VG}| = m \tag{4.14}$$

Assuming equation (4.14) is satisfied, one agent and one virtual goal requires no further comparison for the allocation. Hence:

$$n = 1,\ |\mathscr{VG}| = 1 \Rightarrow T_1 = 0 \tag{4.15}$$

Increasing the number of agents and the cardinality of $\mathscr{VG}$ to two results in:

$$\begin{aligned} n = 2,\ |\mathscr{VG}| = 2 &\Rightarrow T_2 = 3 \\ &= 2^2 + T_1 - 1 \end{aligned} \tag{4.16}$$

Using equation (4.15) and equation (4.16), we get:

$$\begin{aligned} & T_1 = 0 \\ & T_2 = 3 = 2^2 + T_1 - 1 \\ & T_3 = 11 = 3^2 + T_2 - 1 \\ & \qquad \vdots \\ & T_n = n^2 + T_{n-1} - 1 \\ & = n^2 + (n-1)^2 + T_{n-2} - 2 \\ & = n^2 + (n-1)^2 + (n-2)^2 + \cdots - n \\ & = \frac{n(n+1)(2n+1)}{6} - n \\ & = \frac{1}{3}n^3 + \frac{1}{2}n^2 - \frac{5}{6}n \\ & \leq \frac{5}{6}n^3 \leq n^3 \Rightarrow O(n^3) \qquad\qquad n \geq 1 \end{aligned} \tag{4.17}$$

□

4.2 Profile Matrix Permutations Strategy

This algorithm utilizes the profile matrix $\Phi_{n\times m}$ to calculate a coordination strategy where the allocated virtual goals are distinct.[6] The result of the allocation strategy is the assignment of the virtual goals such that the cumulative sum of the votes of robotic agents is maximum. This is achieved through the calculation of the different permutations of the entries of the profile matrix $\Phi_{n\times m}$. These permutations are utilized to ascertain the permutation where the cumulative sum of the votes of the agents is maximum. Moreover, the calculated permutations are required to satisfy the following necessary conditions:

1. Every row of $\Phi_{n\times m}$ is considered in every permutation.

2. Every column of $\Phi_{n\times m}$ is considered exactly once in every permutation.

Condition 1 guarantees all agents are considered in the allocation process. On the other hand, condition 2 ensures that the allocated virtual goals of these agents are distinct. The profile matrix permutations strategy stores these permutations in a matrix $\Psi_{p\times n}$. The number of row entries of this matrix p equals the number of possible permutations of the entries of the profile matrix $\Phi_{n\times m}$. Additionally, the number of columns of this matrix equals the cardinality of the set of virtual goals $|\mathscr{VG}|$. As a result, the best allocation strategy in a decision cycle is the row entry of $\Psi_{p\times n}$ where the cumulative sum of the votes is maximum:

$$\upsilon \in argmax \sum_{i=1}^{p} \sum_{j=1}^{n} \Psi_{ij} \tag{4.18}$$

In equation (4.18), υ is a $1\times n$ vector where every column entry corresponds to an individual agents.

4.2.1 Permutations Calculation

Algorithm 5 shows the process of the calculation of the permutations of the profile matrix $\Phi_{n\times m}$. These permutations are calculated through recursive invocation of *Permutations* function. It finds all the possible permutations of the array $PermuteIndices$ where the size of this array equals the total number of virtual goals. The entries of this array are initialized to zero at the commencement of the computation. These elements represent the

[6] This indicates that $m = n$.

Algorithm 5: Profile matrix permutations through recursive invocation of Permutations(PermuteIndices,k) function.

Data: $\Phi_{n\times m}$, Profile Matrix.
Data: n Number of robots.
Data: m Number of sub-groups ($m = n$).
Data: $PermuteIndices_{1\times m}$ Array of indices, initially all entries initialized to zeros with its length equal m.

```
begin
    static level ← −1;
    static row ← 1;
    level ← level + 1;
    PermuteIndices[k] = level;
    if level == m then
        row ← row + 1;
        for j = 1 : n do
            Ψ[row][j] ← Φ[j][PermuteIndices[j]];
    else
        for i = 1 : m do
            if PermuteIndices[i] == 0 then
                Permutations(PermuteIndices, i);
    level ← level − 1;
    PermuteIndices[k] ← 0;
```

column indices of the profile matrix $\Phi_{n\times m}$. Algorithm 5 passes these values one-by-one to *Permutations* function for every entry of the array that is still zero. Furthermore, it ensures that the elements of $PermuteIndices$ array are not repeated by tracking the entered values through the parameter $level$. After all the elements of the $PermuteIndices$ array are considered in a given permutation (i.e., $level == m$), the new sequencing of the elements of this array is exploited to populate the corresponding row entry of the permutation matrix $\Psi_{p\times n}$ using the votes in profile matrix $\Phi_{n\times m}$.

Next, the permutation matrix $\Psi_{p\times n}$ is used by Algorithm 6 to find the optimal coordination strategy to allocate the virtual goals. It returns the index of the row entry in $\Psi_{p\times n}$ where the cumulative sum of the votes is maximum.

Algorithm 6: Optimal candidate strategy

```
Data: Ψ_{p×n}, Permutation Matrix.
begin
    Max ← −1;
    Index ← −1;
    for i = 1 : p do
        sum ← 0;
        for j = 1 : n do
            sum ← sum + Ψ[i][j];
        if sum ≥ Max then
            Max ← sun;
            Index ← i;
    return Index;
```

4.2.2 Coordination through Permutation: an Example

Consider a team of three robotic agents and a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$. Let $\pi_i(\rho_j)$ represent the vote of i^{th} agent for the virtual goal $\rho_j \in \mathscr{VG}$. This implies that every agent r_i has a vote profile $\Pi_i = \{\pi_i(\rho_1), \pi_i(\rho_2), \pi_i(\rho_3)\}$ where each entry corresponds to the vote of the agent for a virtual goal $\rho_j \in \mathscr{VG},\ j = 1 \dots 3$ (see Chapter 3, Definition 5). Therefore, the profile matrix $\Phi_{3\times3}$ becomes (see Chapter 4, Definition 9):

$$\Phi_{3\times3} = \begin{bmatrix} \Pi_1 \\ \Pi_2 \\ \Pi_3 \end{bmatrix} = \begin{bmatrix} \pi_1(\rho_1) & \pi_1(\rho_2) & \pi_1(\rho_3) \\ \pi_2(\rho_1) & \pi_2(\rho_2) & \pi_2(\rho_3) \\ \pi_3(\rho_1) & \pi_3(\rho_2) & \pi_3(\rho_3) \end{bmatrix} \tag{4.19}$$

The possible permutations of the entries of $\Phi_{3\times3}$ are (see Chapter 4, Algorithm 5):

$$\Psi_{6\times3} = \begin{bmatrix} \pi_1(\rho_1) & \pi_2(\rho_2) & \pi_3(\rho_3) \\ \pi_1(\rho_1) & \pi_2(\rho_3) & \pi_3(\rho_2) \\ \pi_1(\rho_2) & \pi_2(\rho_1) & \pi_3(\rho_3) \\ \pi_1(\rho_2) & \pi_2(\rho_3) & \pi_3(\rho_1) \\ \pi_1(\rho_3) & \pi_2(\rho_2) & \pi_3(\rho_1) \\ \pi_1(\rho_3) & \pi_2(\rho_1) & \pi_3(\rho_2) \end{bmatrix} \tag{4.20}$$

In equation (4.20) every row entry of $\Psi_{6\times3}$ contains a vote value from every robotic agent. Moreover, these votes correspond to distinct columns of $\Phi_{3\times3}$. The optimal allocation strategy where the cumulative sum of the votes of robotic agents is maximum, is calculated as:

$$\begin{aligned} \upsilon \leftarrow MAX(&\{\pi_1(\rho_1) + \pi_2(\rho_2) + \pi_3(\rho_3)\}, \\ &\{\pi_1(\rho_1) + \pi_2(\rho_3) + \pi_3(\rho_2)\}, \\ &\{\pi_1(\rho_2) + \pi_2(\rho_1) + \pi_3(\rho_3)\}, \\ &\{\pi_1(\rho_2) + \pi_2(\rho_3) + \pi_3(\rho_1)\}, \\ &\{\pi_1(\rho_3) + \pi_2(\rho_2) + \pi_3(\rho_1)\}, \\ &\{\pi_1(\rho_3) + \pi_2(\rho_1) + \pi_3(\rho_2)\}) \end{aligned} \tag{4.21}$$

This returns the index of an entry of $\Psi_{6\times3}$ where the cumulative sum of the votes is maximum (see Chapter 4, Algorithm 6). For instance, if equation (4.4) represents the profile matrix associated with the votes of robotic agents r_1, r_2, and r_3 for the set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$, the profile matrix of these agents becomes:

$$\Psi_{6\times3} = \begin{bmatrix} 043 & 0.35 & 0.10 \\ 0.43 & 0.55 & 0.15 \\ 0.17 & 0.10 & 0.10 \\ 0.17 & 0.55 & 0.75 \\ 0.40 & 0.10 & 0.15 \\ 0.40 & 0.35 & 0.75 \end{bmatrix} \tag{4.22}$$

Hence, the optimum allocation using the profile matrix permutations strategy is:

$$\upsilon \leftarrow [\pi_1(\rho_3), \pi_2(\rho_1), \pi_3(\rho_2)] \tag{4.23}$$

This is the last row entry of $\Psi_{6\times3}$ since $0.40 + 0.35 + 0.75 = 1.50$ has the highest cumulative sum of the votes. Therefore, the allocated virtual goals are:

$$r_1 \leftarrow \rho_3,\ r_2 \leftarrow \rho_1,\ r_3 \leftarrow \rho_2 \tag{4.24}$$

Theorem 7. *The profile matrix permutations results in an optimal coordination strategy.*

Proof. Let $\Psi_{p\times n}$ represent the permutation matrix of the robotic agents $r_i,\ i = 1 \ldots n$. Let υ_i denote the cumulative sum of the i^{th} row entry of $\Psi_{p\times n}$. Then, $\mathscr{V} = \{\upsilon_1, \ldots, \upsilon_p\}$ is a set that consists of all the possible cumulative sums of the entries of $\Psi_{p\times n}$. It is apparent that the proof presented in Theorem 4 holds if we replace the vote profile of the agents Π_i in Theorem 4 with $\mathscr{V}$. □

4.3 Discussion

In this Chapter, we introduced two coordination strategies, namely the agents votes maximization and the profile matrix permutations. These strategies are formulated using the vote profiles of the autonomous independent robots.

The agents votes maximization strategy is a heuristic approach to the problem of the coordination of a multi-robot system where the agents with higher votes receive higher priorities. This prioritization is performed dynamically using the votes of the agents at every decision cycle. As a result, it redirects the agents with lower votes to the virtual goals that are least favored during a given decision cycle. These least favored virtual goals are incurred due to their locations with respect to the overall distribution of a set of virtual goals in the environment. In addition, the distances of robotic agents to the virtual goals affect the votes of these goals.

The dynamic prioritization of the agents by the agents votes maximization strategy results in scenarios where some of the agents are not allocated with their best choices of the virtual goals. In equation (4.4) for instance, it is apparent that r_1 ranks the virtual goal ρ_1 with the highest vote $\pi_1(\rho_1) = 0.43$. However, r_1 is assigned to the virtual goal ρ_2 that is ranked with its lowest vote (see equation 4.7). Furthermore, it is possible that the allocation strategy is not optimum in a given decision cycle. Referring to the same equation, the better allocation of the available virtual goals is $r_1 \leftarrow \rho_3, r_2 \leftarrow \rho_1, r_3 \leftarrow \rho_2$ where the cumulative sum of the votes of the agents $0.40 + 0.35 + 0.75 = 1.50$ is maximum.[7] An analysis of $\Phi_{n\times m}$ in this equation reveals that the agents votes maximization strategy selects the second choice of the coordination where the cumulative sum of the votes is $0.17 + 0.55 + .75 = 1.47$ (see equation 4.7). In contrast, the profile matrix permutations calculates the coordination strategy that is optimal (see section 4.2.2 and Theorem 7).

[7] This is the allocation calculated by the profile matrix permutations strategy (see equation 4.24).

The choice of the coordination strategy of a multi-robot system is highly influenced by the specification of a mission and its complexity. The agents votes maximization coordination strategy is unaffected by the number of robotic agents and the cardinality of a set of virtual goals (Lemmas 7 through Lemma 9). On the other hand, the profile matrix permutations strategy is highly dependent on the number of robotic agents as well as the cardinality of the set of virtual goals $\mathscr{VG}$. This dependency is due to the fact that the profile matrix permutations strategy utilizes the possible permutations of the votes of the agents to calculate an optimum allocation at a given decision cycle. Therefore, the rate of the growth of its complexity is exponential to the cardinality of the set of virtual goals, $|\mathscr{VG}| = m$. The approximation of this growth is based on its natural logarithm:

$$\log m! = \sum_{x=1}^{m} \log x \tag{4.25}$$

$$\Rightarrow \int_{1}^{m} \log x \, dx \leq \sum_{x=1}^{m} \log x \leq \int_{0}^{m} \log(x+1) \, dx \tag{4.26}$$

$$\Rightarrow m \log(\frac{m}{e}) + 1 \leq \log m! \leq (m+1) \log \frac{m+1}{e} + 1 \tag{4.27}$$

$$\Rightarrow O(m \log m) \tag{4.28}$$

However, the calculation of the possible permutations of the profile matrix $\Phi_{n \times m}$ where the cardinality of the set of virtual goals is m is:

$$(\frac{m}{3})^m \leq m! \leq (\frac{m}{2})^m, \ \forall m \geq 6 \tag{4.29}$$

Equation (4.29) indicates that the use of the profile matrix permutations strategy to ascertain an optimum allocation is highly expensive when the size of the set of virtual goals is large.

Chapter 5

Multi-Robot Dynamic Multi-Task Allocation

Task-allocation is paramount to the design concept of the cooperation of a multi-robot system. This process facilitates the coordination of the individual robots through the reduction of interference among the agents. For example, it determines the assignments of the individual robots to lifeboats in a sea rescue operation or it specifies the work zones of agents in a foraging mission. Although it is possible to assign these tasks a priori, the predetermined division of labor limits the ability of the system to modify its strategy. However, factors such as the distribution of the tasks, relocations of these tasks or the robotic agents, make it mandatory for a multi-robot system to modify its strategy throughout a mission. Gerkey and Mataric note that research in the field of multi-robot task-allocation is empirical and lacks a formal analysis of these allocation strategies (2004a, p. 939). Furthermore, these strategies do not address the scenarios where the subtasks are dynamic and change their locations continuously (see section 1.3). The rescue mission or the assignments of the targets in a military operation are a few examples where the capability to address the relocations of tasks is crucial for a multi-robot system. In addition, these examples signify a problem domain where the cardinality of the task space is greater than the size of a multi-robot team. As a result, the one-by-one allocation of tasks to the robotic agents is time consuming and inefficient (Dias et al., 2006, p. 1263). These issues underline an open area of research in multi-robot systems that represent an interesting and highly challenging problem domain for further investigation and analysis.

We address some of these issues in a multi-robot, multi-task allocation scenario in this chapter.[1] We utilize the subgrouping decomposition to reduce the task space into a number of subgroups that are equal to the number of robotic agents (see section 2.1). We generate a set of virtual goals using representatives of these subgroups (see section 2.1.2). The agents use these virtual goals to cast their votes for the available subgroups. This reduces the complexity of the decision-making process proportional to the number of agents (see section 2.1.2, Lemma2). We study the effect of the opportunistic ranking module of the external state component of the decision engine of the robotic agents on their votes (see Chapter 3, Definition 7). Furthermore, we utilize the profile matrix permutations strategy (see section 4.2) to coordinate the independent vote profiles of the agents (see section 3.1, Definition 5) for the allocation of the available subgroups.

We use the elapsed time, the distance traveled, and the frequency of the decision-cycle as metrics to analyze the performance of this strategy in contrast to three different coordination strategies. They are the prioritization, the instantaneous, and the time-extended coordination strategies.[2]

The remainder of this chapter is organized as follows. The simulation setup is described in section 5.1. The subgrouping decomposition to generate a set of virtual goals is presented in section 5.1.1. We explore the decision-making of the robotic agents and the effect of the opportunistic ranking module of the external state component of the decision engine on the quality of votes of the agents in section 5.1.2. Sections 5.1.3 and 5.1.4 demonstrate the coordination of the allocation of the virtual goals using the profile matrix permutations strategy. The performance analysis of this strategy is contrasted the prioritization, the instantaneous, and the time-extended strategies in section 5.2. We analyze the effect of the frequency of the decision cycle on the quality of the solution for these strategies in section 5.2.1. The chapter is concluded in section 5.3.

---

[1]We use the term "multi-task" to refer to a task space (see section 2.1.1, Definition 2) that consists of a number of homogeneous physical entities. We refer to a distinct element of this task space by the term "subtask". However, we use this term in a broad and generic sense. In addition, these subtasks are in-motion and change their locations in the environment. Moreover, we assume that the size of this task space is greater than the number of robotic agents.

[2]The prioritization skips the decision-making process through a coordination strategy that allocates tasks a priori. The instantaneous strategy performs the coordination process one time at the commencement of a mission. This coordination is performed at the decision cycles specified by the system, in case of the time-extended strategy. See section 1.3 for the detailed explanation of these approaches.

5.1 Simulation Setup

Figure 5.1 shows a snapshot of the simulation environment. This environment consists of a number of stationary rectangular obstacles. They are depicted in black. These stationary obstacles divide the simulation environment into five regions.

1. **The robotic agents** : There are four robotic agents in every experiment. They navigate the environment with the same constant velocity. Moreover, they interact with their surrounding environment based on their respective simulated on-board sensors.

2. **The task space** : It consists of sixteen dynamic subtasks. They are shown in orange in Figure 5.1. These subtasks exhibit Brownian motion (see Appendix D and Morters and Peres (2010) for further details). Figure 5.2 shows the motion of four of these subtasks during an instance of the simulation.

3. **The decision cycles** : They occur every 200 execution cycles of the simulation. As a result, the subgrouping, the decision-making, and the coordination processes are performed every 200 execution cycles.

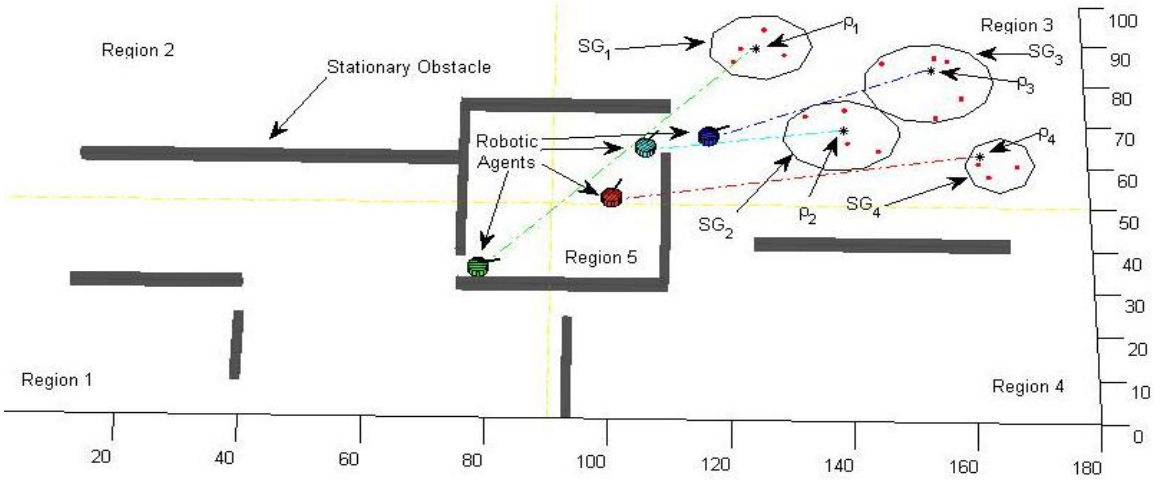

Figure 5.1: Multi-robot dynamic task-allocation. Subgroups are distinguished by the dashed-circles. The subtasks are the red-colored elements enclosed by the dashed-circles of the subgroups. The virtual goals $\rho_j \in \mathscr{VG},\ j = 1 \ldots 4$, of the subgroups are the black-colored asterisks associated with the dashed-circles.

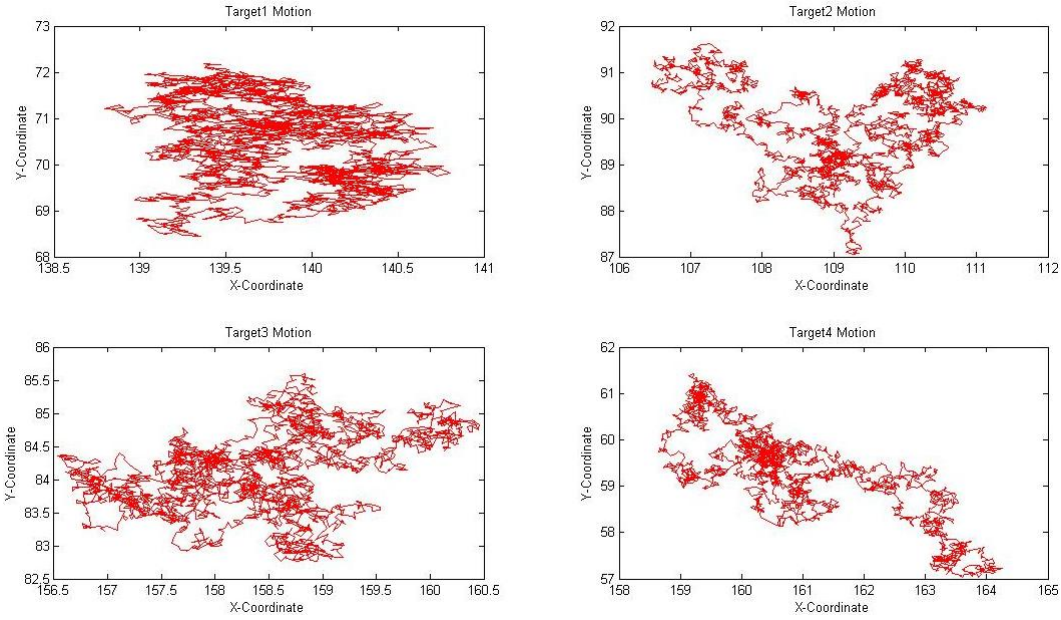

Figure 5.2: The motion of the four of the subtasks during an instance of the simulation.

4. **The subgrouping of the task space** : It is performed at every decision cycle, starting at time $t = 0$. These subgroups are distinguished by the dashed circles in Figure 5.1. We calculate the representatives of these subgroups using equation (2.14). These representatives form a set of virtual goals $\mathscr{VG} = \{\rho_1, \ldots \rho_4\}$.

5. **The voting and coordination** : These are performed at every decision cycle. In Figure 5.1, the colored-links that connect the robotic agents to the virtual goals indicate the allocated subgroups to the agents at the given decision cycle. They are colored after the robotic agents to distinguish between the assignments of the agents.

6. **Finalization of the allocations** : We repeat the subgrouping, the voting, and the coordination processes until one of the robotic agents reaches the virtual goal that represents the allocated subgroup of the agent. We fix the allocated subgroups of these agents thereafter. Subsequently, we provide the robotic agents with the location information for the subtasks of their allocated subgroups after these assignments are fixed. Every agent starts with the subtask of the allocated subgroup that is closest to location of the agent. They proceed to the second closest subtask and so forth until all the subtasks of the allocated subgroups are attended.

7. **The specification of the mission** : We relocate the task space to another region of the simulation environment after the agents attend all the subtasks of their allocated subgroups. The relocation of the task space begins the next instance of the mission. We repeat this procedure until the task space is relocated in all the five regions of the simulation environment (see Figure 5.1). As a result, every mission consists of five instances that are followed continuously.

5.1.1 Subgrouping of the Task Space

The decision-making and consequently the coordination of a multi-robot system are complicated and cumbersome processes if the size of the task space is larger than the size of the robotic team. These processes become more inefficient and time-consuming when the subtasks change their locations continuously. This is due to the fact that any inference on the future allocation of subtasks demands estimates of the individual agents on the entire task space. Furthermore, these estimates become outdated and unreliable once the subtasks change their locations in the field of the operation.

The task subgrouping process (see section 2.1) provides the means to address these issues. This process decomposes a mission into a number of subgroups that are equal to the number of the robotic agents. Furthermore, these subgroups are populated with an approximately equal number of subtasks (see Chapter 2, Algorithm 1 and Algorithm 2). The subgrouping process reduces the interference among the robotic agents. More specifically, it allocates robots to disjoint subgroups and hence they do not share a common workload.

Figure 5.3 shows the process of the subgrouping where the task space is decomposed into four subgroups. In the figure, the subplot (1) shows the task space before the application of the subgrouping. The result of the subgrouping of these subtasks is presented in subplot (2). The subgroups are colored red, green, blue and cyan.

The subgrouping process facilitates the decision-making and the coordination of a multi-robot system through the introduction of a set of virtual goals that are representatives of the subgroups (see Chapter 2.1.2, Definition 3). Subplot (2) in Figure 5.4 shows the representatives by asterisks of the same color of their respective subgroups. In this subplot, the subtask $\tau_{(i,j)}$ is the i^{th} element of the j^{th} subgroup $\tau_{(i,j)} \in \mathscr{SG}_j$ (see Chapter 2, Definition 2). The representatives form the set of virtual goals $\mathscr{VG} = \{\rho_1 \dots \rho_4\}$ (see Chapter 2, Definition 1). They are used by the robotic agents to vote the available subgroups.

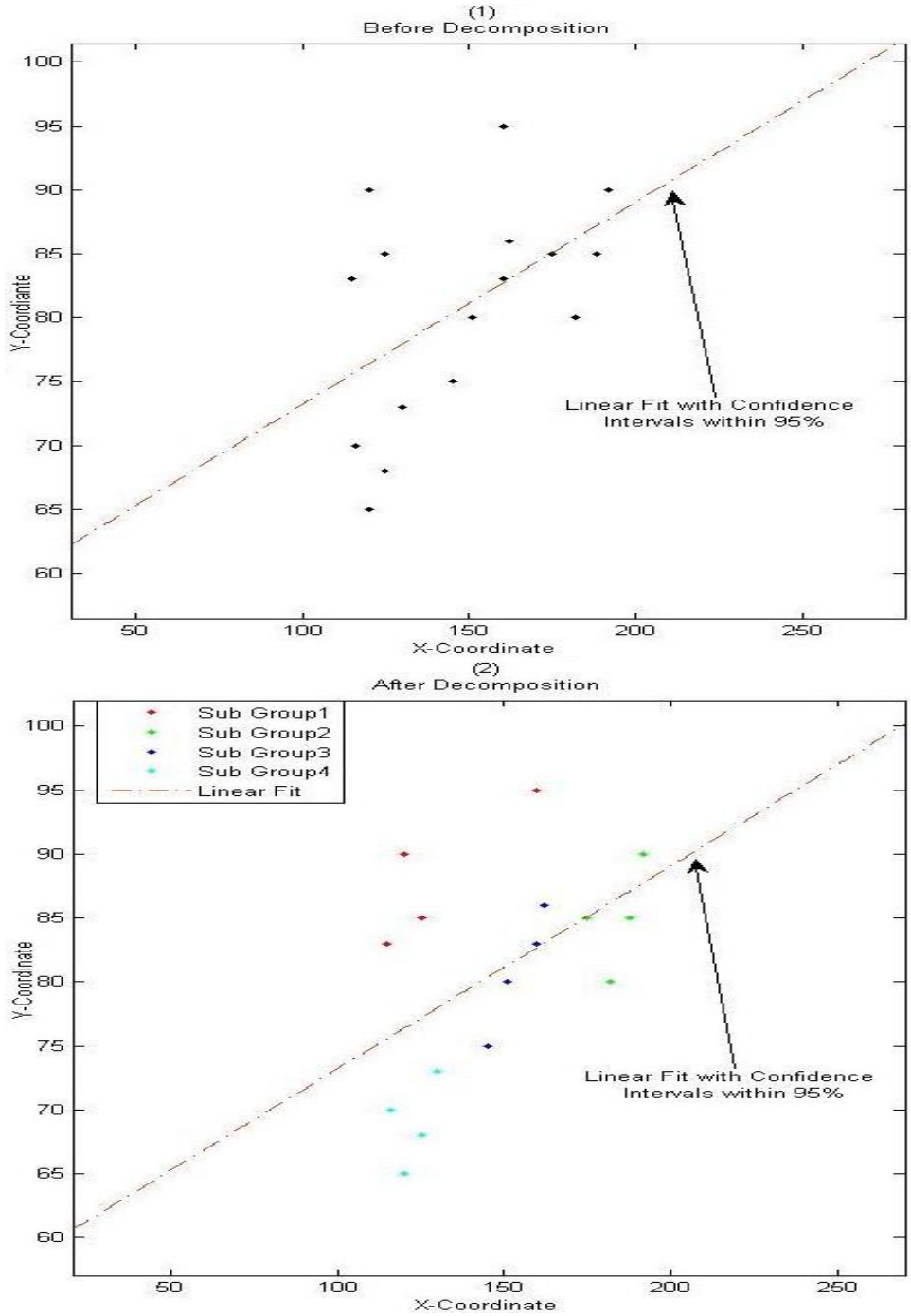

Figure 5.3: Subgrouping of the task space $\mathscr{T}$ into four disjoint subgroups. (1) The original task space before subgrouping. (2) The resulting four subgroups after subgrouping. A linear fit with the confidence intervals within 95% is provided for comparison.

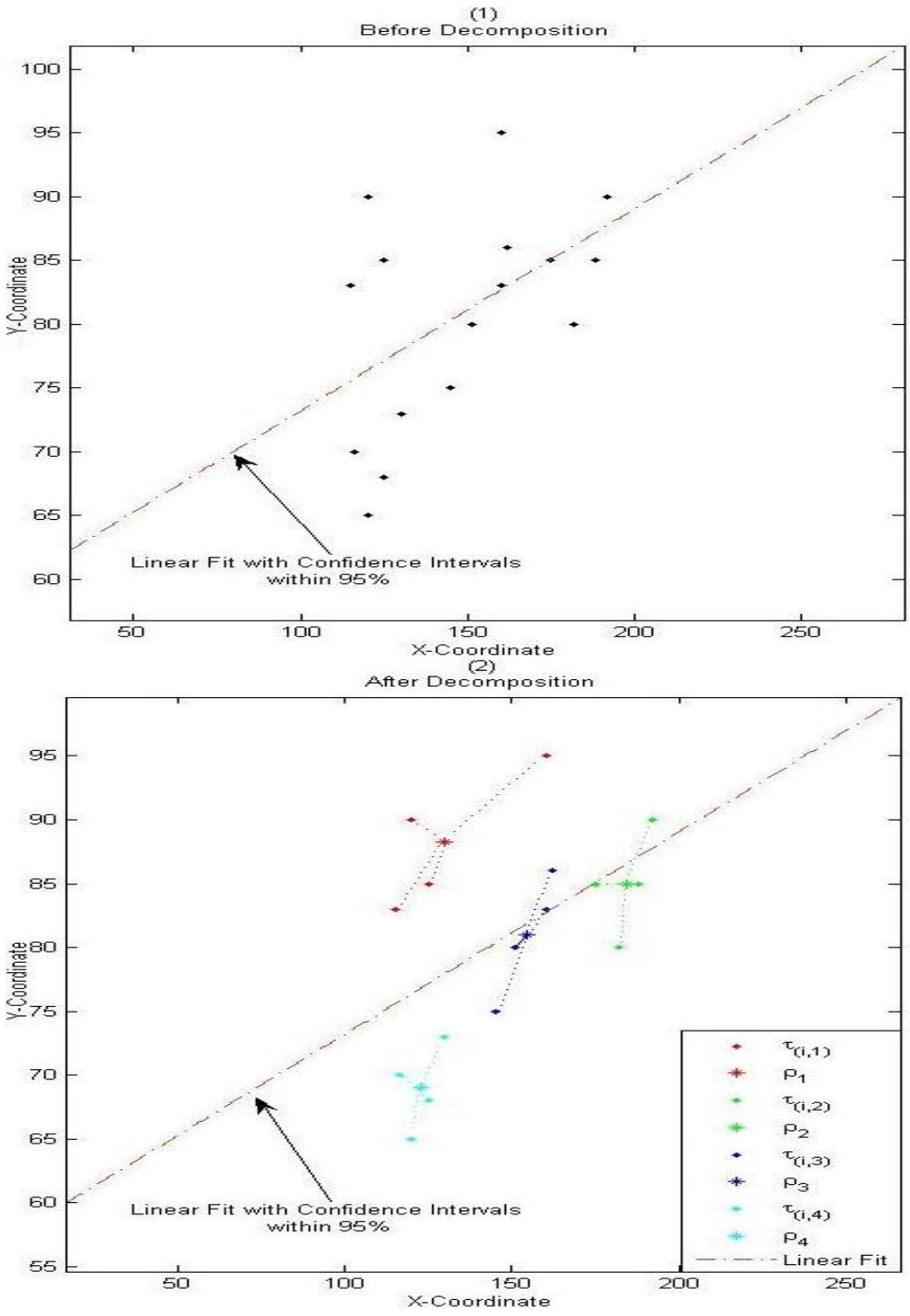

Figure 5.4: Subgrouping of the given task space $\mathscr{T}$ into four disjoint subgroups. (1) The original task space before subgrouping. (2) The resulting subgroups same as last. Asterisks ρ_i are the virtual goals of the subgroups. A linear fit with the confidence intervals within 95% is provided for comparison.

5.1.2 Voting of the Robotic Agents

The representation of the task space in terms of a set of virtual goals reduces the voting effort of the robotic agents by a factor that is proportional to the size of the robotic team (see section 2.1.2, Lemma 2). Figure 5.5 through Figure 5.8 show the results of the voting of the agents during one of the instances of the simulation. Figure 5.5 and Figure 5.6 correspond to the default ranking module of the external state component of the decision engine (see section 3.2, Definition 6). The performance of the opportunistic ranking module of the external state component of the agents is shown in Figure 5.7 and Figure 5.8 (see section 3.2, Definition 7).

Figure 5.5 shows that the votes of the agents do not evolve and remain the same (almost 0.25) for more than 2500 execution cycles. The constant ranking of the virtual goals implies that the robotic agents interpret all the subgroups with equal confidence. This neutral ranking behavior of the default ranking module starts to change when the distances of the agents to the virtual goals decrease. However, these ranking values exhibit a highly fluctuating and irregular behavior. Figure 5.5 evinces this peculiar behavior of the votes of the default ranking module by the abrupt rises and the falls of the corresponding curves of these values. These delays of the evolution of the votes is further illustrated in Figure 5.6.

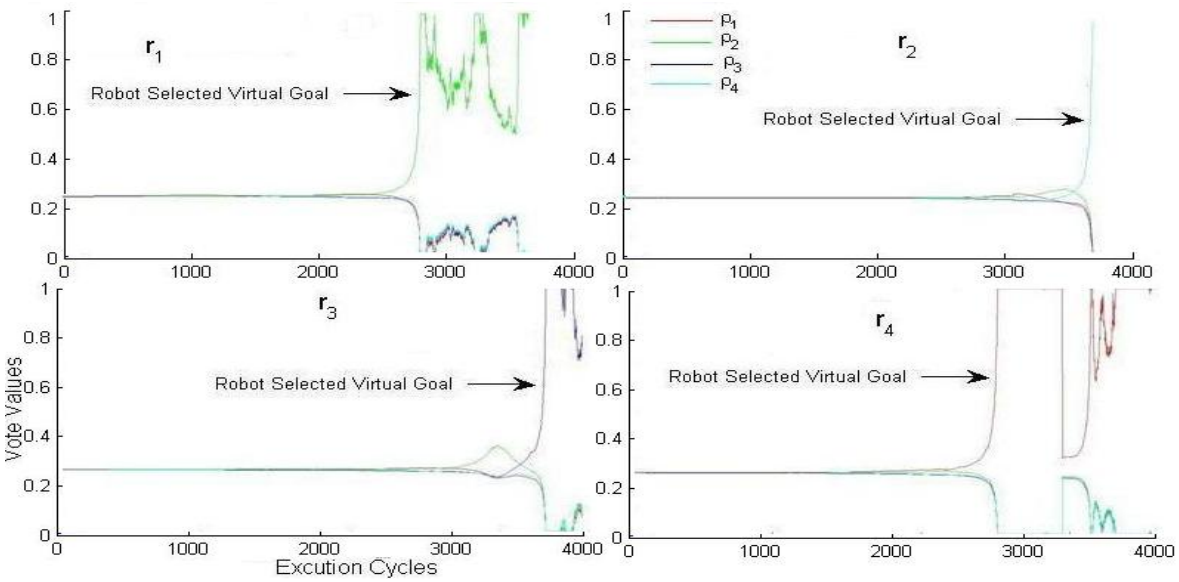

Figure 5.5: The evolution of the votes of the default ranking module $\pi_i^t(r_i \mapsto \rho_j)$ of the robotic agents.

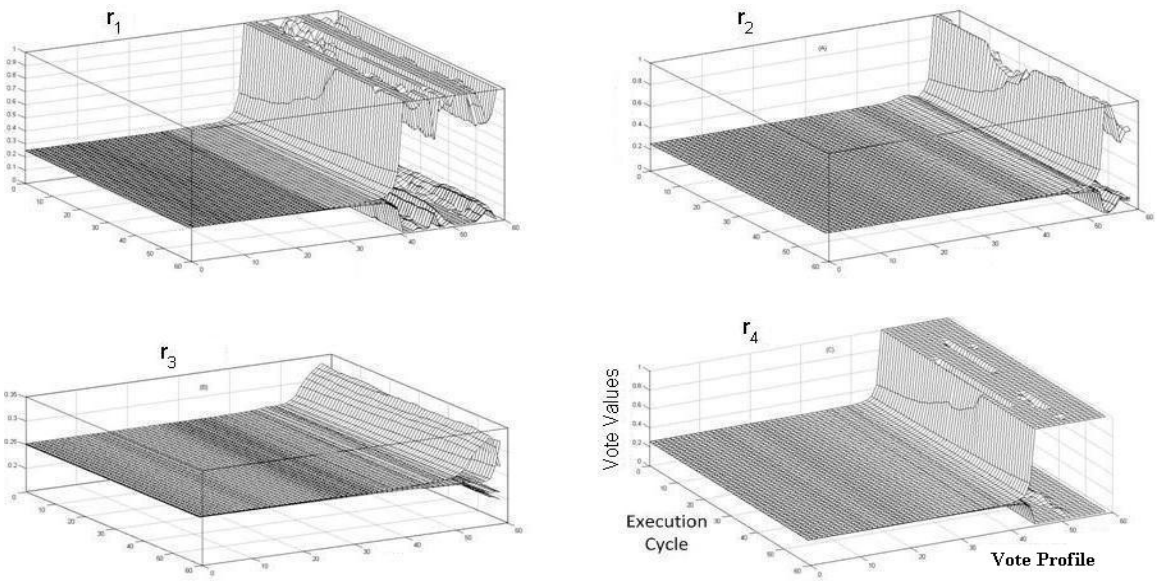

Figure 5.6: The evolution of the default ranking module $\pi_i^t(r_i \mapsto \rho_j)$ of the robotic agents.

The opportunistic ranking module of the external state component of the decision engine amends this undesirable behavior. This module regulates the votes of the default ranking module of the decision engine by incorporating an incremental evolution of the votes of agents (see Figure 3.1). It provides the robotic agents with concise information of their confidence on different virtual goals between the consecutive decision cycles. In particular, it enables the agents to retrospect their decisions through the introspection of their ranking of the virtual goals at every execution cycle.

The formulation of the opportunistic ranking module comprises an initialization parameter where the value of this parameter is on $[0 \dots 1]$ interval (see equation 3.7).[3] We use $C = 0$ throughout the experiments to prevent any *a priori* information. More specifically, we avoid any prioritization or biased initialization of the ranking of the virtual goals. This is achieved through the incorporation of the opportunistic ranking module that relies solely on the votes that are acquired during the operation.

Figure 5.7 and Figure 5.8 show the results of the ranking of the virtual goals after the incorporation of the opportunistic raking module into the decision engine. It is apparent that the decision mechanism exhibits higher flexibility in voting on these virtual goals after

[3]This parameter is used primarily to prevent any unexpected behavior of the decision engine due to an undefined value of the opportunistic ranking module at the first decision cycle.

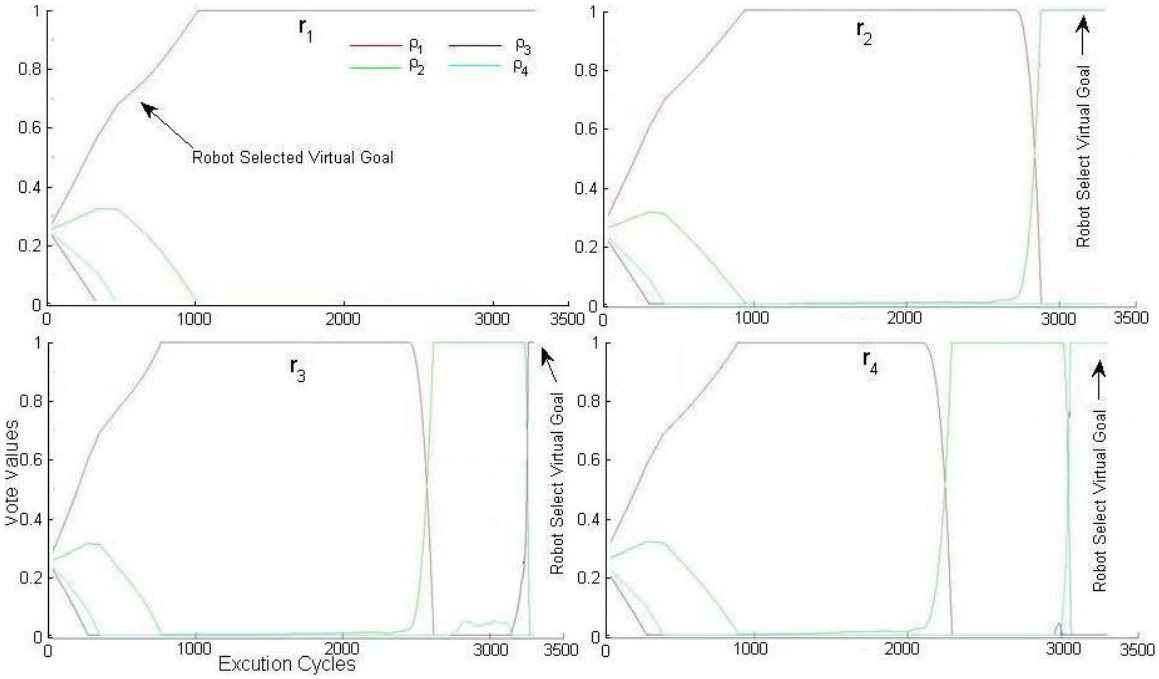

Figure 5.7: The vote values after the incorporation of the opportunistic ranking module $\omega_i(\rho_j)$, with $C = 0$ in equation (3.9).

the introduction of the opportunistic module. For instance, this module enables the agents to independently estimate their confidence on available virtual goals. (see Chapter 3, Definition 8 and Theorem 4). Figure 5.7 indicates that the robotic agents r_2, r_3, and r_4 change their elected virtual goals after 2100 execution cycles. However, this manipulation of the best choice for a virtual goal progresses smoothly. In other words, the opportunistic ranking module enables the decision mechanism to distinguish between the overlapping choices of the virtual goals that are voted closely (see Chapter 3, Theorem 6). This is evident in the modifications of the best choices of the virtual goals of the robotic agents r_2, r_3, and r_4 after 2100 execution cycles in Figure 5.7. The effect of the incorporation of the opportunistic ranking module of the external state component of the decision engine on the final votes of the robotic agents is further illustrated in Figure 5.8.

5.1.3 Profile Matrix Permutations Strategy

Although the decision engine of the robotic agents enhances their ability to maintain the best choices of virtual goals autonomously, it does not advocate the evolution of cooperation

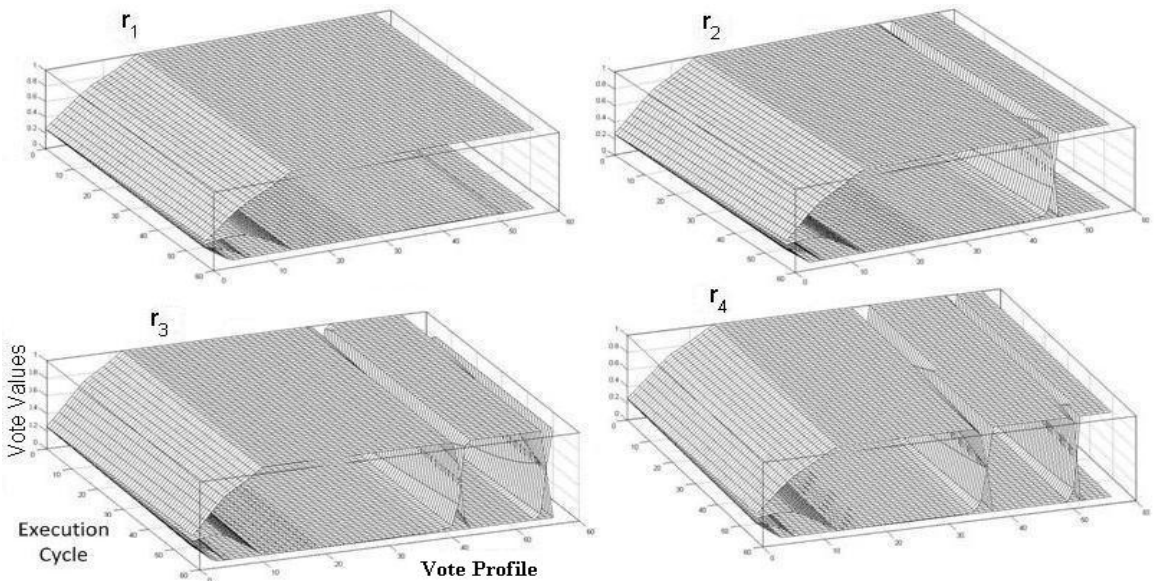

Figure 5.8: The vote values after the incorporation of the opportunistic ranking module $\omega_i(\rho_j)$, with $C = 0$ in equation (3.9).

at the group-level. Figure 5.7 reveals that all robotic agents elect the virtual goal ρ_1 (i.e., the red-colored curve in this figure) as their best choice for more than 2200 execution cycles. Between 2500 and 3000 execution cycles the trend of the overlapping choices of the virtual goals continues among r_2, r_3, and r_4 that elect ρ_2 (i.e., the green-colored curve in Figure 5.7). This results in a situation where a majority of the subtasks of the overall task space are unattended if the robots are allowed to solely follow their independent decisions. However, Figure 5.1 shows that these agents are assigned to different virtual goals. In addition, the separation of the allocation of the virtual goals is evident in Figure 5.7. In this figure, the final allocated virtual goals of r_1, r_2, r_3, and r_4 are ρ_1, ρ_2, ρ_3, and ρ_4, respectively. These are the red-, green-, blue-, and cyan-colored curves.

We achieve this coordination of the allocation of the virtual goals through the application of the profile matrix permutations strategy (see section 4.2, Algorithm 5). This strategy utilizes the profile matrix (see Chapter 4, Definition 9) of the vote profiles of robotic agents (see Chapter 3, Definition 5) to select the votes where the cumulative sum of the votes are maximized (see section 4.2.2).

Figure 5.9 shows the result of the computation of the possible permutations of the votes.[4]

[4]There are 24 permutations of the votes.

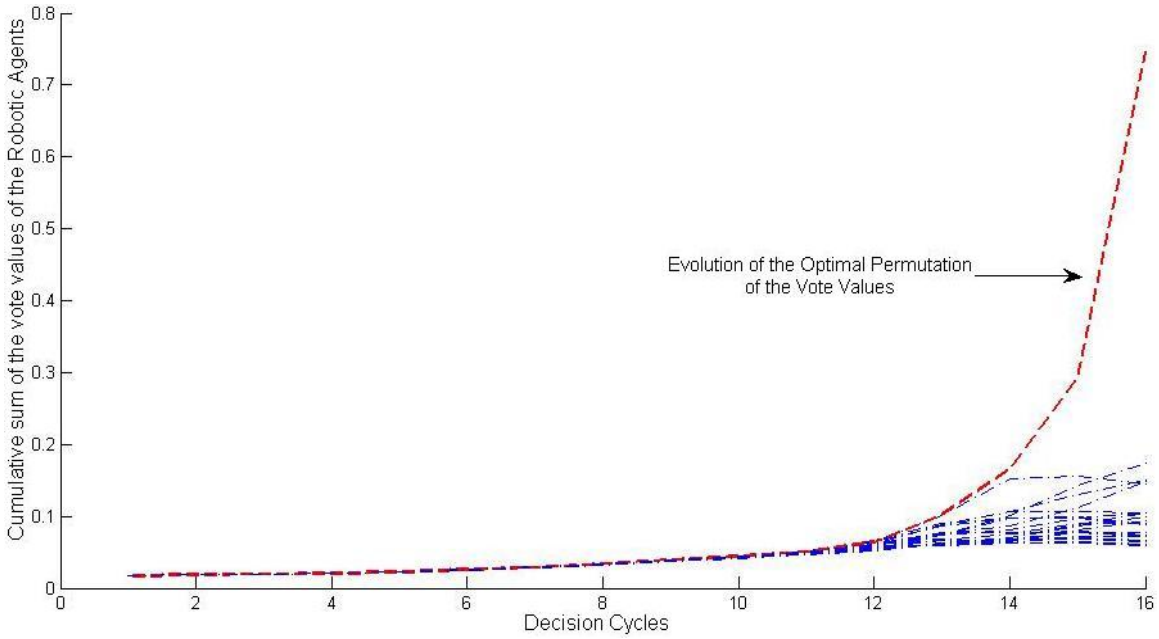

Figure 5.9: The evolution of the selected permutation (red-colored curve) along with other possible permutations of the vote values of the robotic agents.

In this figure, the red-colored curve is the dominant strategy where the cumulative sum of the votes of robotic agents is maximum. In contrast, the blue-colored curves are the rejected permutations of the votes. These are the permutations where the cumulative sum of the votes is persistently below the cumulative sum of the votes acquired by the dominant strategy. Figure 5.10 signifies the result of this dominant allocation strategy in contrast to the other possible permutations. This figure demonstrates that the selected strategy optimizes the allocation of the virtual goals to the robotic agents (see Chapter 4, Theorem 7). This optimum strategy has a direct influence on the distances traveled during a mission. This is due to the fact that the decision engine of the robotic agents considers the distances of these agents to the available virtual goals to compute the default ranking of the votes (see equation 3.4). Figure 5.11 verifies that the allocation strategy where the cumulative sum of the votes is maximum improves the total distance traveled at the group-level. Although this improvement is negligible during the first eight decision cycles, the difference between the distances traveled based on the selected strategy and other possible permutations increases thereafter. This increase of the difference of the distance traveled based on different permutations is illustrated in Figure 5.12. This figure shows that the confidence interval of

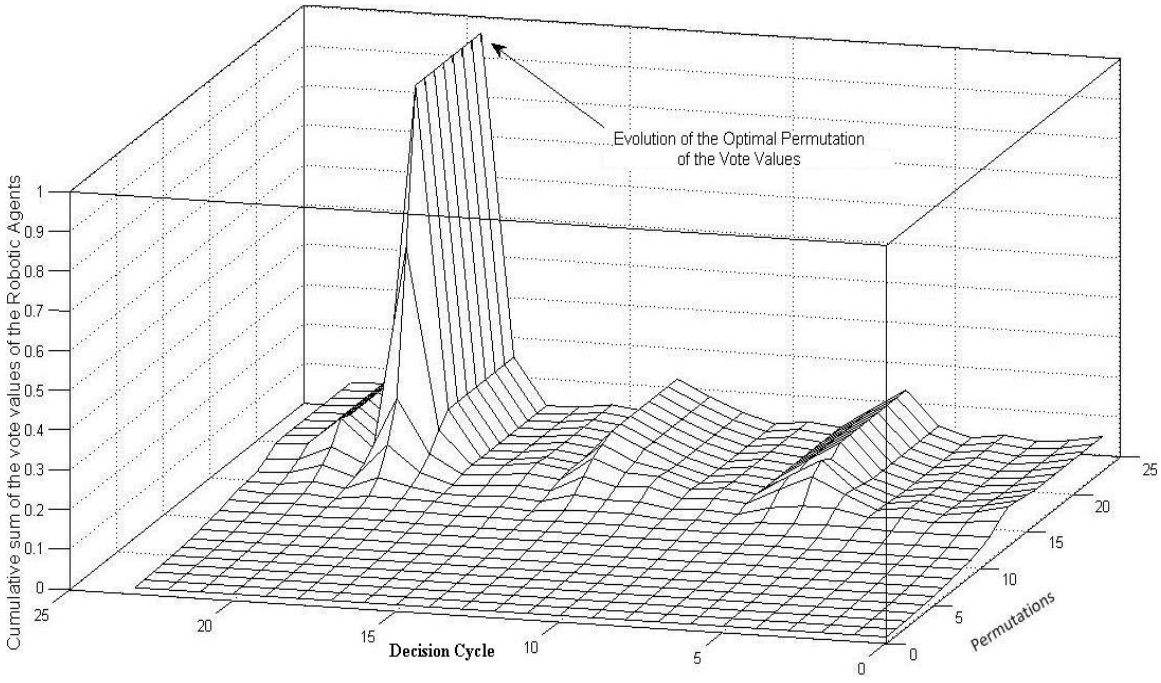

Figure 5.10: The evolution of the optimal allocation strategy in contrast to the other possible permutations.

the distance traveled by the robotic agents using the selected allocation strategy steadily improves throughout the mission. The increase of the difference of the distance traveled at the group-level based on the selected strategy and other possible permutations is apparent.

Figure 5.13 illustrates the process of the allocation of the virtual goals using the profile matrix permutations strategy. The execution cycles are presented along the x-axis of this figure. The entries of the y-axis correspond to the available virtual goals. They are ρ_1, ρ_2, ρ_3, and ρ_4, respectively. The sequences of the colored squares that elongate along the x-axis are the indicators of the allocated virtual goals of the agents at a given execution cycle. They are colored red, green, blue, and cyan to denote the assignment of the robotic agents r_1, r_2, r_3, and r_4. However, this coloring scheme is independent of the adapted scheme in Figure 5.5 and Figure 5.7 that represents the specific virtual goals. The double-headed arrows in this figure indicates the timespan of the allocation of the specific virtual goal to a robotic agent.

Figure 5.13 shows that the best choices for the virtual goals of some of the agents are not compromised to achieve the optimum allocation strategy at the group-level. In particular,

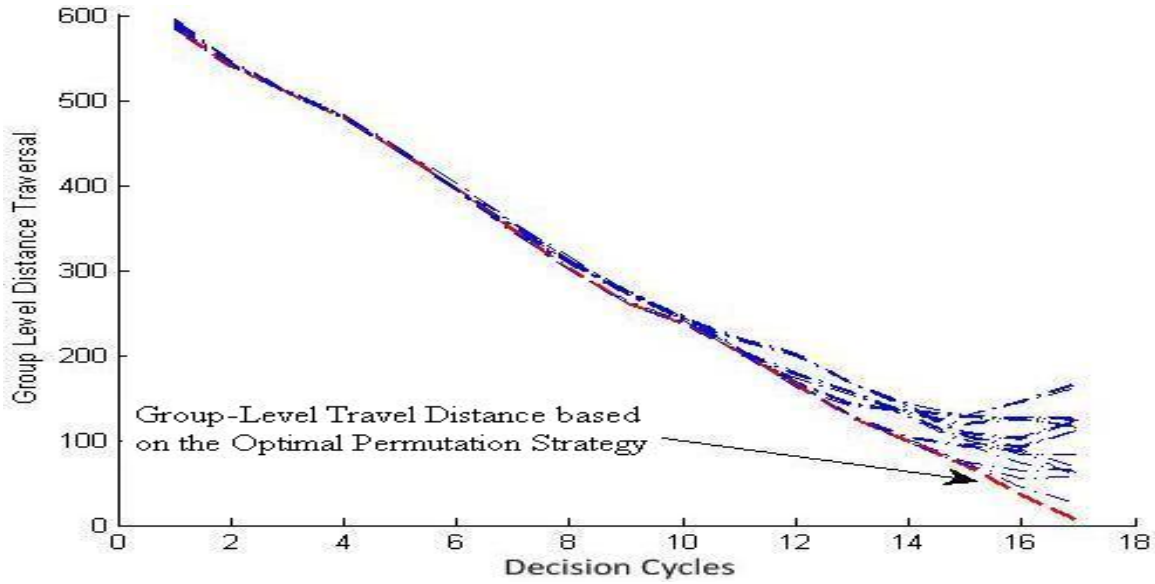

Figure 5.11: The total distance traveled by the robotic agents at the group-level based on different permutations of the vote values of robotic agents.

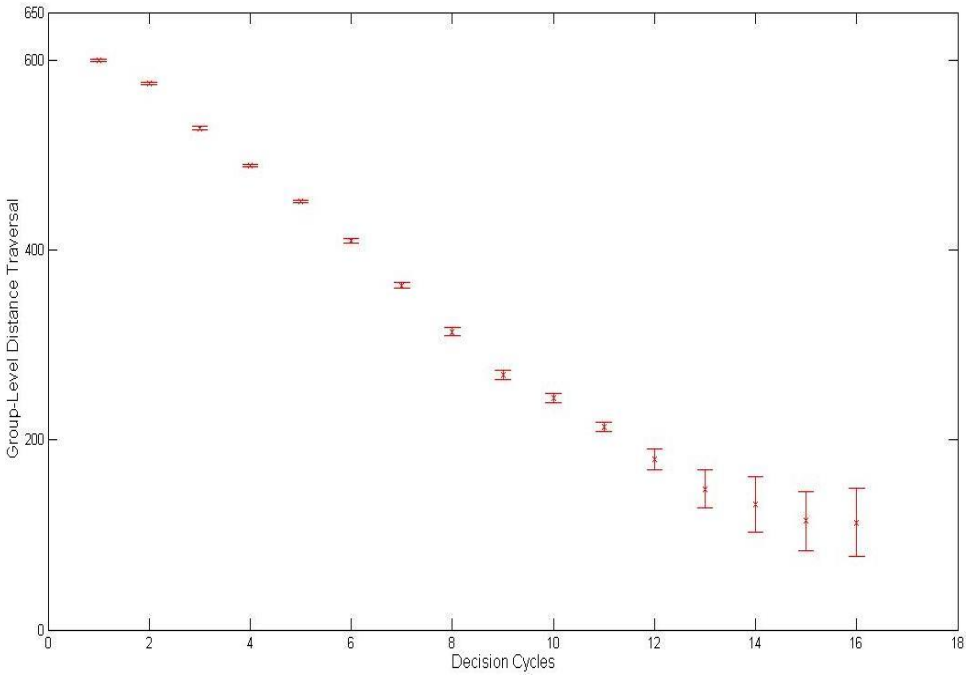

Figure 5.12: The confidence interval of the travel distance of the optimal allocation strategy at the group-level in contrast to the other permutations.

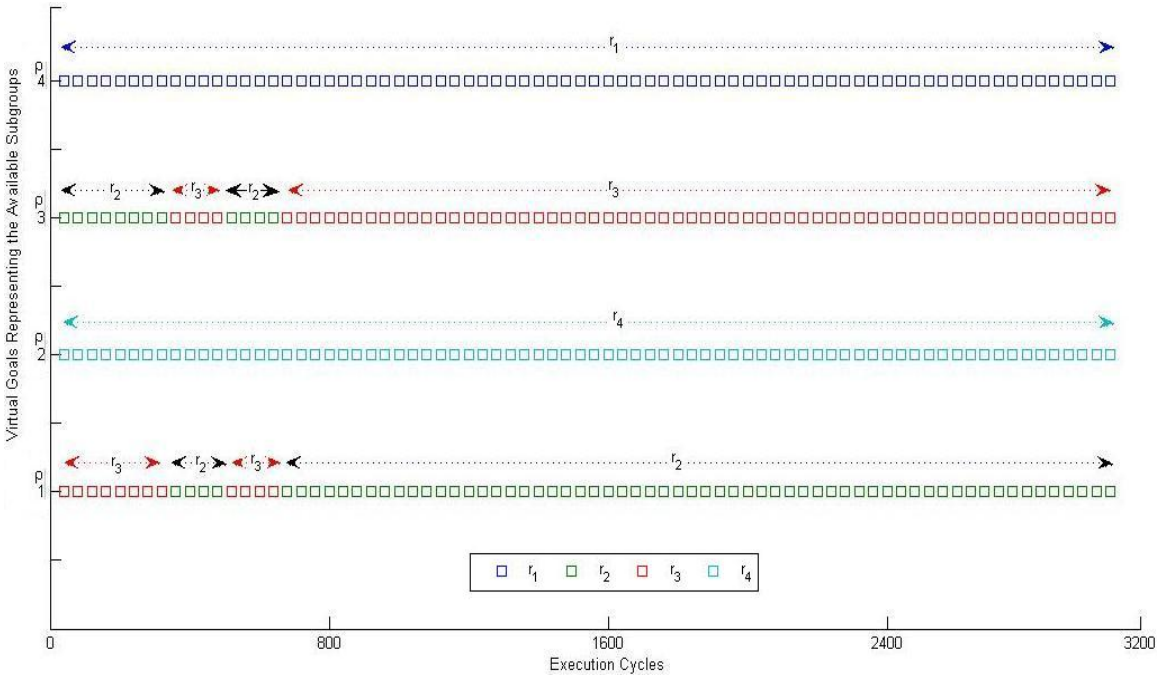

Figure 5.13: The evolution of the allocated virtual goals of the robotic agents at different execution cycles.

r_1 follows its best choice of the virtual goal ρ_4 throughout the mission. This is verified by the comparison of the allocated virtual goal of r_1 in Figure 5.13 and the result of the decision mechanism of this agent in Figure 5.7. This demonstrates the capability of the decision engine to elect the best choice of the virtual goal of the agents at the individual-level in a given decision cycle (see Chapter 3, Theorem 4). The allocated virtual goal of r_4 is also unaltered by this allocation strategy. However, the allocated virtual goal for r_4 (ρ_2) is not the best choice of the agent when compared with the result of the decision mechanism of r_4 in Figure 5.7. The profile matrix permutations strategy selects a permutation where the cumulative sum of the votes is maximum. This optimal strategy manipulates the choice of r_4 to follow the virtual goal that is dominated primarily by the best choices of this agent. These are virtual goals ρ_4 and ρ_2. This is verified through the comparison of the entry of r_4 in Figure 5.13 and the decision mechanism of r_4 in Figure 5.7. In contrast, the allocated virtual goals of r_2 and r_3 are highly influenced by the performance of the profile matrix permutations strategy. Figure 5.13 manifests this influence through the alternating behavior of the assignments of r_2 and r_3 between the virtual goals ρ_1 and ρ_3, respectively.

5.1.4 Effect of the Appearance of Obstacles on the Allocation Strategy

An interesting aspect of the profile matrix permutations coordination strategy is the change in behavior of the permutations in the advent of unexpected events in the environment. Figure 5.14 presents a scenario where the robotic agents encounter the presence of obstacles along their paths. These encounters compel the robotic agents to change their direction to avoid collisions with the obstacles. The changes in the navigational paths of agents affect the votes for the available virtual goals that are calculated by the default ranking module of the decision engine. Figure 5.14 shows the effect of the occurrences of the obstacles on the distance traveled at the group-level. In this figure, every peak in the curves of the permutations corresponds to the occurrence of an obstacle. Figure 5.15 illustrates the effect of the obstacles on the evolution of the selected allocation strategy in contrast to other possible permutations. It is apparent that the obstacles result in the decline of the cumulative sum of the votes of the robotic agents in all the permutations.

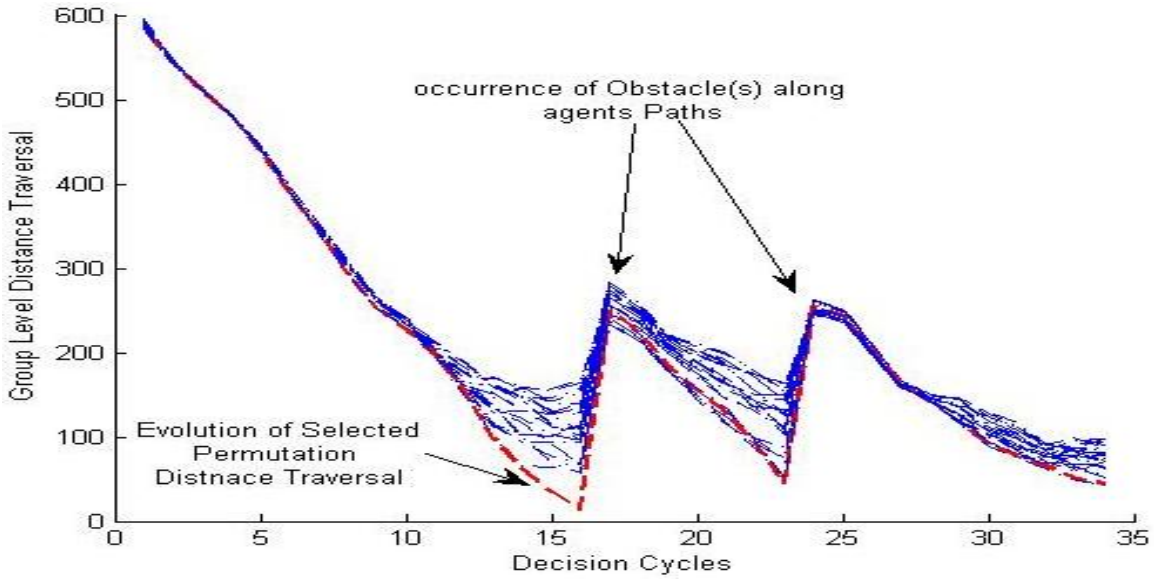

Figure 5.14: The effect of the occurrences of the obstacles on the group-level travel distance.

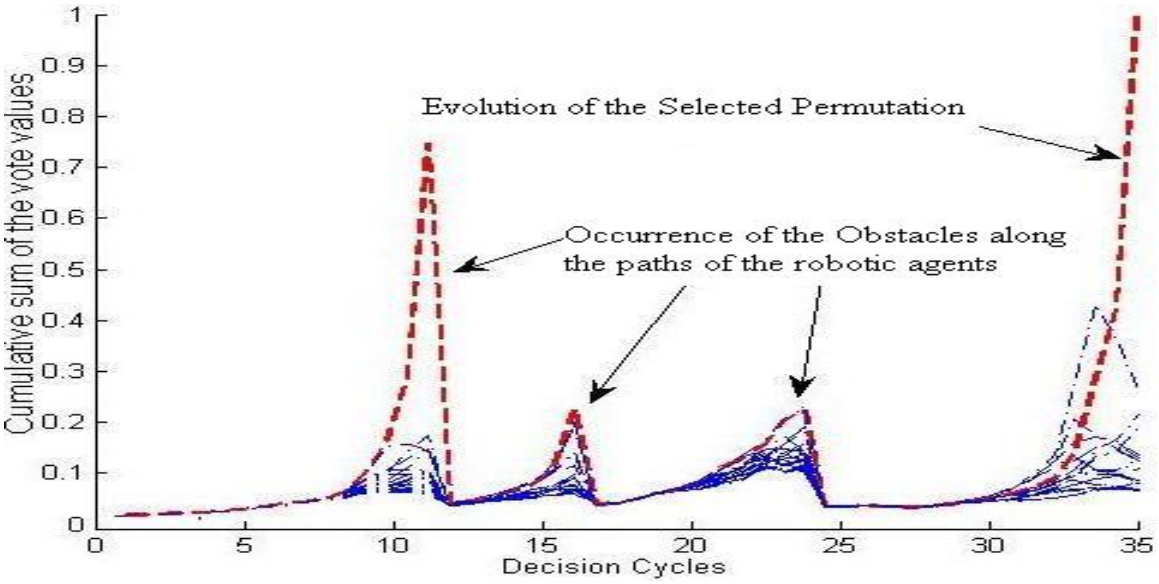

Figure 5.15: The evolution of the optimal allocation strategy (i.e., red-colored curve) in the presence of obstacles.

5.2 Further Analysis

This section examines the performance of the profile matrix permutations coordination strategy in comparison to the prioritization, the instantaneous, and the time-extended allocation strategies. We consider the elapsed time, the distance traveled and the frequency of the decision cycles to evaluate the performance of these strategies. The prioritization strategy implies a fixed allocation of the available virtual goals to robotic agents at the commencement of a mission. The agents vote for the virtual goals in the instantaneous and the time-extended strategies. These votes are cast only once at the commencement of a mission or at the specific decision cycles for the instantaneous and the time-extended strategies, respectively. We use the Hungarian algorithm (see Appendix D and Kuhn, 1955) to coordinate the votes in the instantaneous and the time-extended coordination scenarios (see Chapter 4, p. 72 for details).[5] We use the same location information of the robotic agents to study the performance of these strategies. Furthermore, we use the same decision mechanism to calculate their votes (see section 3.1). Moreover, we initialize the task

[5]Liu and Shell (2011) introduce an interval-based version of the Hungarian algorithm. However, we do not use the interval-based version of the Hungarian algorithm in the context of the analysis of this dissertation.

space in the environment using the same location information. However, the behavior of the subtasks slightly varies in different experiments. This is due to the randomness that is a characteristic of Brownian motion. The layout of the environment remains the same for all these strategies. In addition, we consider a scenario where an entire mission comprises five instances (see section 5.1).

Figure 5.16 shows the distance traveled in contrast to the number of execution cycles. This figure represents the completion of an instance of a mission using the prioritization, the instantaneous, the time-extended, and the profile matrix permutations allocation strategies. The results in Figure 5.16 indicate that the prioritization and the profile matrix permutations strategies achieve a shorter travel distance by the robotic agents compared to the instantaneous and the time-extended strategies. In particular, this difference is evident in the performance of the instantaneous strategy. The prioritization and the profile matrix permutations complete this instance of a mission in almost 3300 execution cycles. However, it takes approximately 3800 and 4800 execution cycles for the time-extended and the instantaneous allocation strategies to complete the same instance of a mission. Although all these strategies exhibit the same decreasing pattern in the travel distance in first 1000

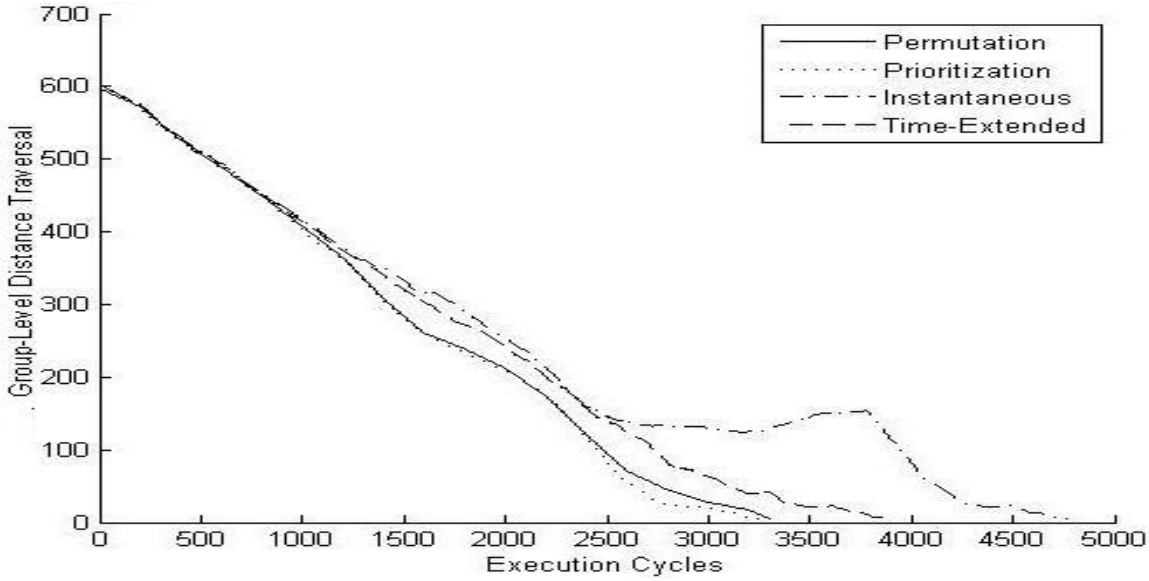

Figure 5.16: The travel distance in comparison to the number of the execution cycles to complete an instance of a mission using the prioritization, the instantaneous, the time-extended, and the profile matrix permutations allocation strategies.

execution cycles, the gap between the distances imposed on the robotic agents increases thereafter. Figure 5.16 reveals that the profile matrix permutations and the prioritization strategies exhibit a relatively similar decreasing pattern of the travel distance. However, the prioritization allocation strategy shows a slightly faster decreasing pattern of travel distance between 2500 and 3300 execution cycles compared to the profile matrix permutations.

Table 5.1 provides the means and the standard deviations of the elapsed time and the travel distance for the robotic agents using the profile matrix permutations, the prioritization, the instantaneous, and the time-extended strategies in a complete mission (see section 5.1). The travel distance entry of this table indicates that the profile matrix permutations strategy achieves the smallest mean travel distance at the group-level to complete a mission. Furthermore, the standard deviation of the distances traveled in Table 5.1 demonstrates the differences of the mean in travel distances using different strategies. The standard deviation of the distances traveled by these agents using the perioritization, the instantaneous, and the time-extended allocation strategies are not within the one standard deviation of this distance using the profile matrix permutations strategy. This result verifies the capability of the profile matrix permutations to compute an allocation strategy that is optimal (see Chapter 4, Theorem 7).

The instantaneous strategy expends the least time to complete a mission. Although the time-extended strategy performs better than the prioritization strategy, this strategy is outperformed by the profile matrix permutations and the instantaneous strategies. This outcome is supported by the results of the travel distance and the elapsed time of the time-extended strategy in Table 5.1.

Table 5.1: The mean and the standard deviation of the elapsed time and the travel distance where the decision-making and the coordination processes are performed every 200 execution cycles.

Allocation	Elapsed Time(sec.)		Travel Distance	
Strategies	Mean	STD	Mean	STD
Permutation	48.96	29.82	294.26	121.71
Prioritized	52.84	13.74	364.54	156.86
Instantaneous	45.73	39.10	326.50	128.65
Time-Extended	50.10	35.59	345.36	131.24

5.2.1 Effect of the Change of the Decision Cycles

Table 5.1 presents scenarios where the decision cycle of the system is set to 200 execution cycles. This setting of the decision cycle has a substantial influence on the performance of the system. In particular, the decision cycle defines the frequency of the calculation of the subgroups and their respective virtual goals. Therefore, the frequency of the decision-making and the coordination processes are dependent on the setting of the decision cycle in the system. The decision-making and the coordination are performed once at most in the instantaneous and the prioritization strategies.[6] However, the robotic agents equipped with these strategies require updates on the locations of the virtual goals. This information is provided at every decision cycle.

Table 5.2 and Table 5.3 present the effect of the change of the decision cycle on the performance of these strategies. Specifically, these tables show the means and standard deviations of the elapsed time and the travel distance of the robotic agents where the decision-making and the coordination processes are performed every 100 and every 300 execution cycles, respectively. A comparison of the entries of Table 5.1 through Table 5.3 denotes an improvement of the distance traveled by the robotic agents using the profile matrix permutations strategy. Furthermore, these results verify that the reduction of the travel distance is proportional to the frequency of the decision cycle. In other words, the distance traveled using the profile matrix permutations strategy decreases with the reduction of the frequency of the decision cycle. However, in the strategy there is a trade-off between the decrease in distance and elapsed time. More specifically, the profile matrix permutations

Table 5.2: The mean and standard deviation of the performance of the strategies where the decision-making and the coordination processes are performed every 100 execution cycles.

Allocation	Elapsed Time(sec.)		Distance Traversal	
Strategies	Mean	STD	Mean	STD
Permutation	40.57	31.64	326.51	131.64
Prioritized	42.60	28.56	325.48	127.03
Instantaneous	50.42	29.68	300.29	121.84
Time-Extended	49.93	41.44	338.51	138.58

[6]The prioritization does not involve any decision-making. It coordinates the allocation of the virtual goals to the robotic agents at the commencement of a mission preemptively.

Table 5.3: The mean and standard deviation of the performance of the strategies where the decision-making and the coordination processes are performed every 300 execution cycles.

Allocation	Elapsed Time(sec.)		Distance Traversal	
Strategies	Mean	STD	Mean	STD
Permutation	51.12	34.75	293.05	118.72
Prioritized	40.42	29.60	317.39	125.16
Instantaneous	48.53	27.03	397.20	168.34
Time-Extended	52.51	19.75	382.15	151.10

strategy achieves an improvement in the travel distance at the cost of the longer completion time of the mission.

On the contrary, the instantaneous and the time-extended strategies exhibit an increase in the mean travel distance of the robotic agents that is inversely proportional to the frequency of the decision cycle. The mean values of these strategies indicate that the travel distance increases when the frequency of the decision cycle decreases. In addition, the mean values of elapsed time of these strategies follow the same trend. However, the elapsed time is less affected by the reduction in the frequency of the decision cycle compared to the profile matrix permutation strategy.

The mean and standard deviation of the travel distance and the elapsed time of the prioritization strategy do not exhibit a particular increasing or decreasing pattern with respect to this change of the frequency of the decision cycle. As a result, any conclusion on the influence of the frequency of the decision cycle on the performance of this strategy is not warranted.

Figure 5.17 presents the confidence intervals for the travel distance of the profile matrix permutations, the prioritization, the instantaneous, and the time-extended strategies presented in Table 5.1 through Table 5.3.

5.3 Discussion

Search and rescue, and catastrophic area exploration and evacuation are among the problem domains that benefit from this research. These problems represent a class of applications where the overall mission of a multi-robot system is composed of several subgoals. Furthermore, the dynamic allocation of these subgoals is a crucial factor that influences the

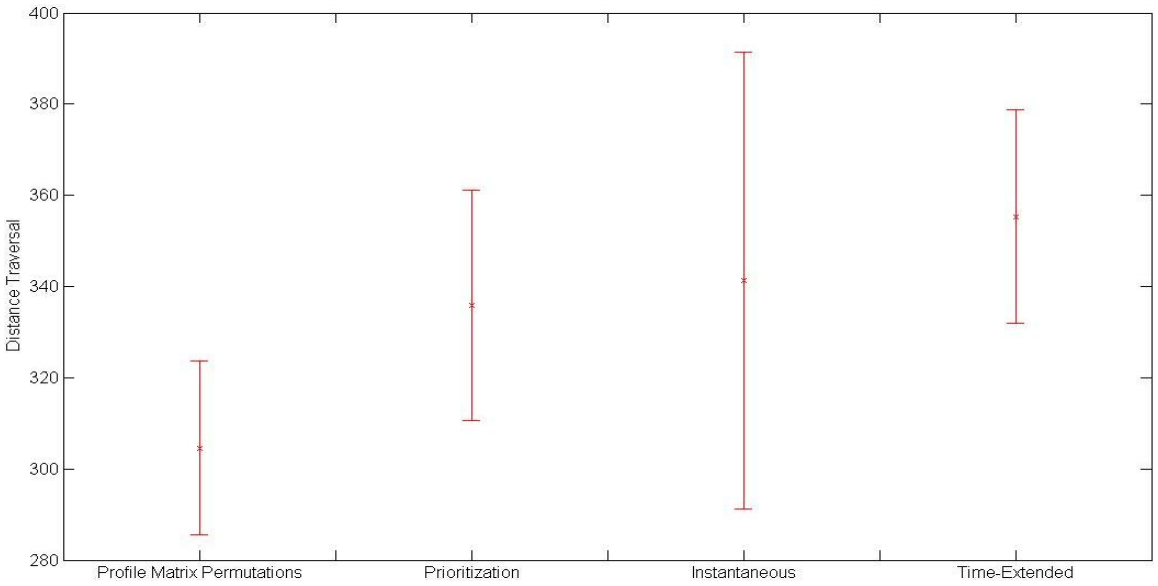

Figure 5.17: The deviation of the distance traveled by the robotic agents using the profile matrix permutations, the prioritization, the instantaneous, and the time-extended strategies from the mean in response to the change of the frequency of the decision cycle of the system.

efficiency of the performance of a multi-robot system. This is due to the fact that these subgoals change their locations in the environment over time. For instance, in a sea rescue mission the lifeboats change their locations in response to the changing current. Moreover, in catastrophic area exploration and evacuation the affected people intentionally move to search for possible escapes routes.

Brownian motion is a natural choice to modeling of the behavior of the subtasks in a rescue and evacuation scenarios. This model derives the displacement of the subtasks from independent and identically distributed random variables at every execution cycle. As a result, the motion of these subtasks exhibits nondeterministic behavior.

The scale of the problem is another factor that significantly affect the overall performance of the system. For instance, a rescue mission comprises of hundreds subgoals. Dias et al. note that the computation of an optimum allocation strategy where the robots vote on all possible combinations of the subtasks is exponential in the size of the task space (Dias et al., 2006, p. 1263). In addition, the available approaches to subgrouping the task space (see section 2.1 for the review of this topic) compromise the performance of these approaches

(ibid. p.1263). However, the results presented in this chapter demonstrate a reliable approach to subgrouping of the task space that preserves the quality of the solution to the decomposition of the overall mission.

Dias et al. state that the reduction of the problem of the outdated solution to a local search problem where the neighboring robots trade their allocated tasks, faces the issue of the local optima (Dias et al., 2006, p. 1264). The partial observability of the environment, the varying costs of the operation, and the change in the location information of the subtasks are among reasons that raise the issue of the outdated solution of a strategy. However, we demonstrate that the combination of the subgrouping and the profile matrix permutations strategy provides a potential approach to address the issue of outdated solutions. Furthermore, this combination provides system with the ability to allocate subtasks to the robotic agents in a scale that is significantly larger than one-to-one mapping of the robots and the subtasks.

There exists a number of ongoing issues and considerations that require further investigation and analysis. The decision mechanism of the robotic agents in this study is not constrained by parameters, such as level of acuteness or the preemptive preferences among the subtasks. This simplification of the decision-making and subsequently the coordination of a multi-robot system is not reflected in many real-life scenarios where the ordering and the speed of the transition among subtasks significantly affect the result of the operation.

Moreover, computation of all the possible permutations of the vote values of the robotic agents to infer the optimum allocation is a limitation of the profile matrix permutations strategy. The division of the task space into a number of subgroups expedites the decision-making and the allocation processes for small- to mid-sized multi-robot systems. However, the complexity of this coordination strategy to infer the optimum allocation increases exponentially as the size of the system grows. Research in the field of multi-robot task allocation indicates that the growth of the complexity of the available allocation strategies is proportional to the number of the subtasks and robotic agents (see Nanjanath and Gini, 2010, p. 903, and Dias et. al, 2006, p. 1265, for further details). More specifically, the complexity of these strategies are bounded between $O(nm^2)$ and $O(n^4)$ where m and n represent the size of the task space and the number of the robotic agents, respectively.[7] This observation suggests that the complexity of the profile matrix permutations can be improved through

[7]The upper boundary $O(n^4)$ corresponds to the scenarios where $m = n$.

the introduction of a subgrouping mechanism that allocates subgroups to the clusters of the robotic agents. For instance, it is apparent that the complexity of the profile matrix permutations is less than the available allocation strategies when $n \leq 5$ where n is the total number of the robotic agents.

The incorporation of a learning process that enables the profile matrix permutations strategy to eliminate the cumulative sum of the votes that are persistently below a threshold is another alternative to improve the complexity of this strategy. The introduction of a number of mediators that perform the allocations in parallel and in a distributed fashion is also an interesting and challenging problem that can be studied in the future.

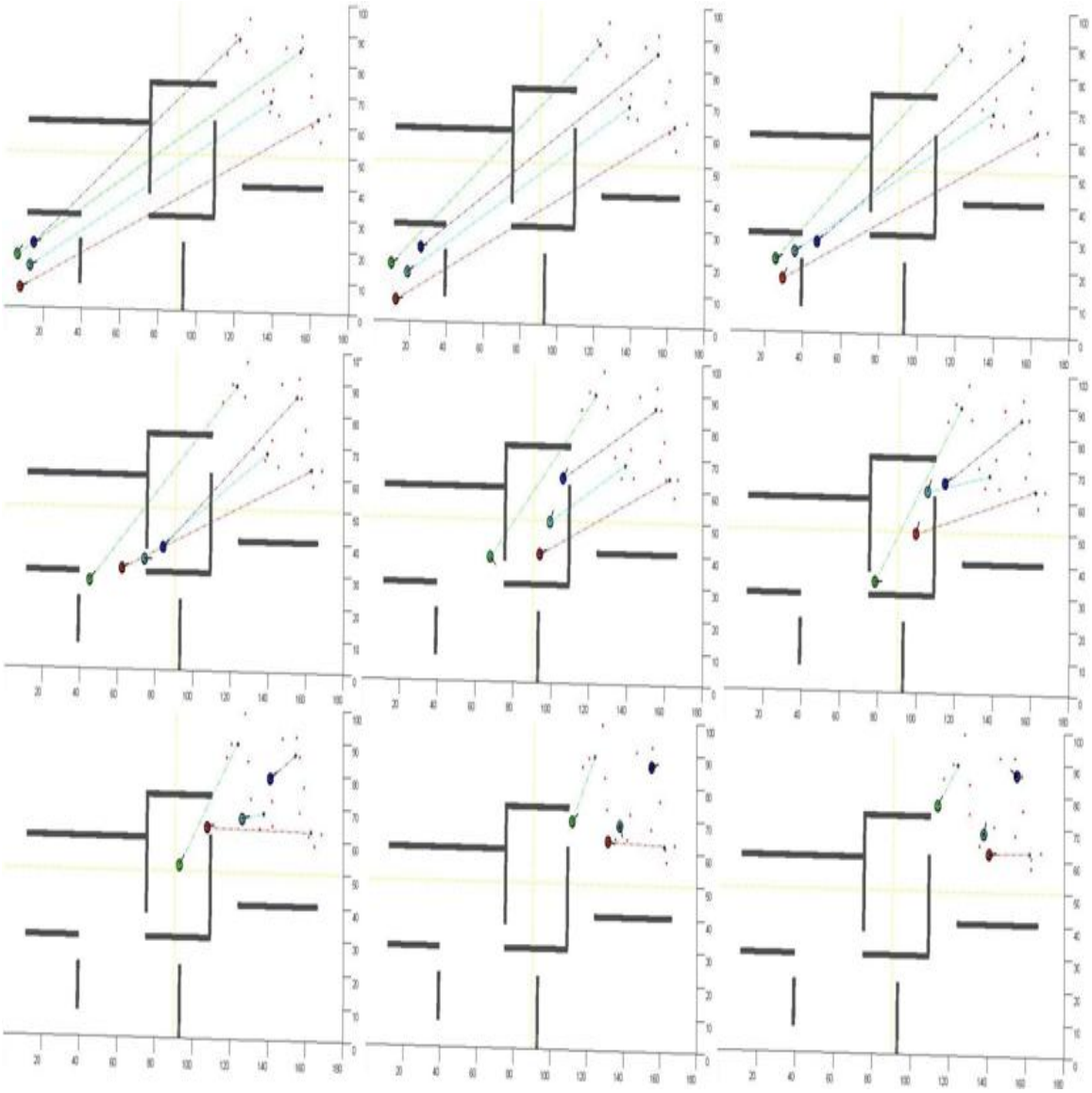

Figure 5.18: Multi-robot, dynamic multi-task allocation. The assignments of the robotic agents to the virtual goal of a subgroup are indicated by a line that connects a robotic agent to a given virtual goal. These lines are colored same as the agents to distinguish between their allocations.

Chapter 6

Multi-Robot Multi-Location Rendezvous

We studied the performance of a multi-robot system in a dynamic multi-task allocation scenario in Chapter 5. We elaborated the application of the subgrouping to decompose a mission into a number of subgroups and their representatives. Furthermore, we demonstrated the capability of profile matrix permutations to acquire an optimum strategy to allocate these subgroups to robotic agents. This strategy is calculated using the votes of agents where the cumulative sum of the votes is maximum.

The distributions of the robotic agents in the environment is valuable information to the decomposition process. The role of this information is paramount in scenarios where a multi-robot system requires participation in repetitive tasks in a timely basis. These tasks vary from the congregation of individual agents in specific facility locations (e.g., Dudek and Roy 1997; Cortes 2006; Lin and Anderson 2003) to logistic scenarios where the robots assemble in a varying numbers to receive certain types of services such as maintenance, repairs, battery exchange (e.g., Ngo et al. 2008), and recharging.

In this chapter, we study a generic rendezvous problem where a group of robotic agents rendezvous with a special purpose service robot.[1] We utilize the location information of the agents to decompose the rendezvous mission into a set of virtual goals using the ORD and

[1]We use the term service in its very broad and general sense as opposed to a particular type of service. However, we assume a type of service where two or more robots physically meet in order to exchange a quantity such as energy or spare parts as a service. Furthermore, we assume one of these robots is the sole service provider robot (see Zebrowski and Vaughan (2005) for an example).

the LSRD decomposition techniques (see section 2.2 and Appendix A). These virtual goals form the rendezvous locations of the robotic agents.

We study the performance of the decision engine of the robotic agents to rank the virtual goals (i.e., rendezvous locations). We use the agents votes maximization strategy to coordinate the allocation of the rendezvous locations (see section 4.1). In addition, we analyze the performance of the system where a given virtual goal is constrained with its number of assignments (see Chapter 4, Definition 10).

We consider two types of simulation environments. They are the obstacle-free environment and the environment that comprises a number of stationary obstacles. We use these environments in conjunction with the travel distance of the robotic agents as the metric to analyze the performance of our approach in comparison to the fixed service station (e.g., Silverman et al. 2002; Oh and Zelinsky 2000) and the single dynamic rendezvous location (e.g., Zebrowski et al. 2007).

The remainder of this chapter is organized as follows. Section 6.1 describes the simulation setup. We present the process of the generation of virtual goals using the ORD and the LSRD in section 6.2. Sections 6.3 and 6.4 provide details on the performance of the decision engine of the individual agents and the agents votes maximization coordination strategy in the absence and the presence of obstacles. We provide analysis of the performance of the system in section 6.5. We conclude this chapter in section 6.6.

6.1 Simulation Setup

We study a multi-robot, multi-location rendezvous scenario in two environmental setups. They are an obstacle-free environment and an environment that comprises a number of stationary rectangular obstacles. In addition, we consider two types of the robotic agents in the simulations. They are the worker robots and a single service robot. They interact with the surrounding environment using their respective on-board simulated sensors. Furthermore, they perform simple reactive collision avoidance to avoid the obstacles and the robotic agents in their vicinity. Additionally, these agents from a homogeneous multi-robot system.

1. **The worker robots** : They represent a group of six autonomous robotic agents that are deployed in a working environment and rendezvous with the service robot. Furthermore, these worker robots participate in a rendezvous mission jointly (i.e., the

rendezvous mission is performed at the group-level).

2. **The service robot** : This robotic agent resembles a single, mobile support unit with the capability of relocation in the environment. We assume that the service robot is the sole service-provider unit for the worker robots. Furthermore, we assume this service robot is self-contained and does not require service.

3. **Virtual goals generation** : We utilize the location information of the worker robots, the ORD, and the LSRD to decompose the rendezvous mission into a set of virtual goals $\mathscr{VG}$ (see section 2.2 and Appendix A). These virtual goals represent the rendezvous locations of the worker and service robots. Furthermore, we consider the effect of the cost of relocations of the worker robots on the computation of these virtual goals (see equation 2.16, and equation A.9 through equation A.11).

4. **The distributions of the worker robots** : We consider the following distributions of the worker robots in the simulations:

   - The worker robots are located closest to each other and farthest from the location of the service robot.
   - The worker robots are located as close as possible to the location of the service robot.
   - The worker robots are located arbitrarily in the environment.
   - The worker robots are located the farthest distance from the location of the service robot.

5. **The placement of the service robot** : We locate the service robot in the top and in the bottom left most and right most corners as well as the center of the environment. This is done to evaluate the effect of the distributions of the worker robots on the rendezvous process.

6.2 Virtual Goals Generation

Figure 6.1 and Figure 6.2 represent the set of virtual goals generated using the ORD (see Chapter 2, Algorithm 3) and the LSRD (see Appendix A) decompositions, respectively. Red-colored circles represent the locations of the worker robots. The location of the service

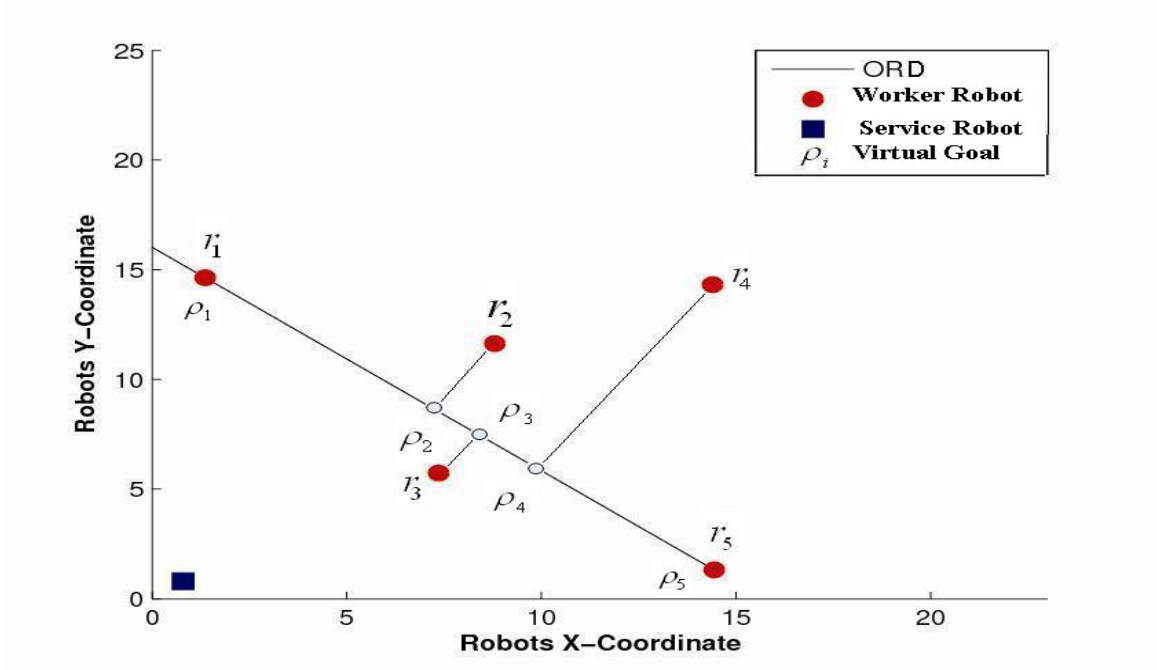

Figure 6.1: The generated set of virtual goals using the ORD decomposition. Virtual goal $\rho_i \in \mathscr{VG}$ corresponds to the i^{th} worker robot.

robot is represented by the blue-colored square at the bottom left corner of these figures.

Figure 6.1 verifies that the virtual goals of the collinear agents r_1 and r_2 coincide with their corresponding locations (see Chapter 2, Lemma 3). Moreover, this figure illustrates the locations of at least two of the virtual goals along the route that passes through the set of virtual goals (see Chapter 2, Lemma 5). It is also apparent that the locations of these virtual goals are at the interceptions of the normals from the locations of the worker robots to this route (see Figure 2.2 and equation 2.22). Furthermore, this route divides the worker robots into two sets of equal sizes (see equation 2.17).

Appendix A shows that the cost of the relocation of the worker robots influences the generation process for a set of virtual goals using the LSRD decomposition. Additionally, it describes the weighting criteria that express the cost of the relocation of worker robots to their respective virtual goals. Figure 6.2 illustrates the effect of these costs on the generation of the set of virtual goals. In this figure, the set of virtual goals that correspond to a specific weighting criterion are represented by similar symbols. For example, the virtual goals generated using $w_i = \frac{1}{y_i}$ as the cost of relocation of worker robots are represented by

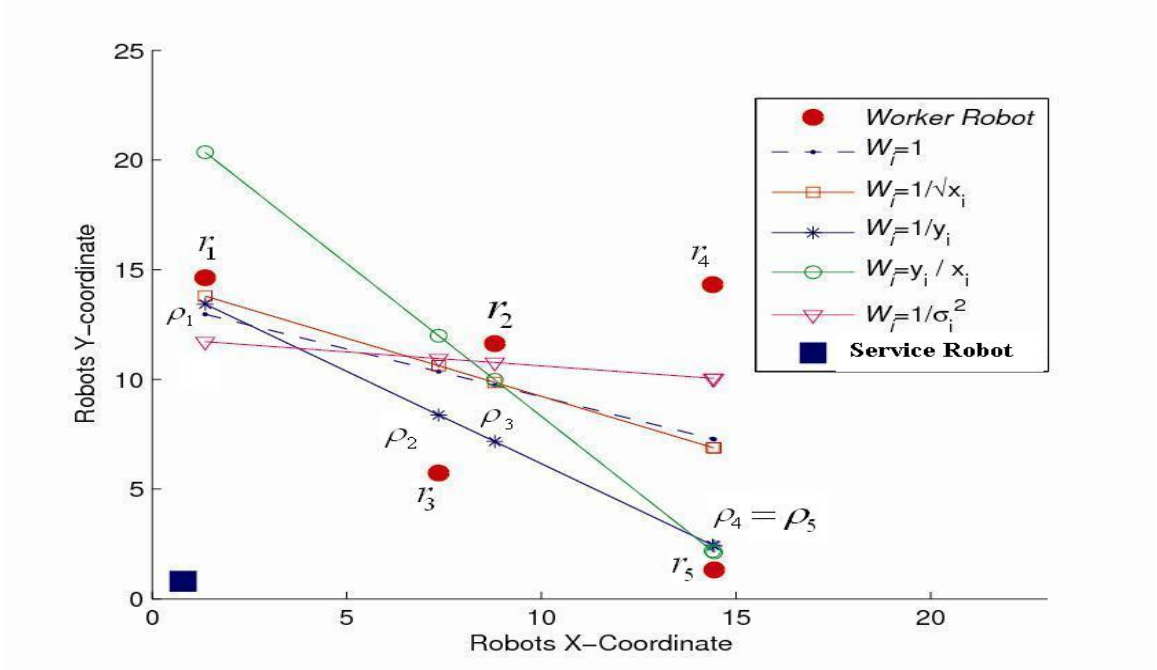

Figure 6.2: The generated set of virtual goals using the LSRD decomposition. Only the virtual goal $\rho_i \in \mathscr{VG}$ that corresponds to $w_i = \frac{1}{y_i}$ is labeled in the figure.

the blue-colored asterisks. Furthermore, these virtual goals are connected by lines that show the collinearity among members of the same set of virtual goals. For instance, the collinearity of the set that corresponds to the weighting criterion $w_i = \frac{1}{y_i}$ is shown by the blue-colored line that passes through the locations of the blue-colored asterisks.

Figure 6.2 shows that the generated set of virtual goals exhibits different alignments, in response to the adapted weighting criterion, to calculate the cost of the relocation of the worker robots. Every change in the weighting criterion of the cost of the relocation of worker robots results in the generation of a different set of virtual goals. The values of these weighting criteria are highly dependent on the distributions of worker robots in the environment. For example, the weighting criterion $w_i = \frac{1}{\sigma_i^2}$ induces the locations of the virtual goals to be closer to the locations of the worker robots where the variation of the y-coordinate of these robots is less. Regardless of the weighting criterion of their respective set of virtual goals, all goals that are generated based on the location information of a specific worker robot are vertically aligned. This is the result of the assumption of the y-coordinate of the worker robots as the independent variable for the formulation of the

LSRD decomposition (see Appendix A and section 2.2.1 for further explanation).

6.3 Obstacle-Free Environment

Figure 6.3 shows a snapshot of the simulation in an obstacle-free environment where a group of six worker robots are engaged in a multi-robot, multi-location rendezvous mission. The service robot is depicted in cyan. The green-colored worker robot r_1 indicates that the rendezvous of the agent with the service robot is completed. In contrast, agents that are moving towards their corresponding virtual goals to complete their rendezvous are shown in yellow. Figure 6.3 corresponds to the scenario where the ORD decomposition is utilized to generate the set of virtual goals. These virtual goals are the locations where the worker robots rendezvous with the service robot. They are labeled $\rho_i, i = 1 \ldots 6$, and depicted by the red-colored asterisks. The one-to-one correspondence between the worker robots and their respective virtual goals is apparent in this figure (see section 2.2.2 and Appendix A for further explanation). Figure 6.3 shows that the ORD decomposition generates a set of virtual goals where the collinearity of these virtual goals divides the worker robots into two sets of robots. This is verified by the number of worker robots above and below the ORD

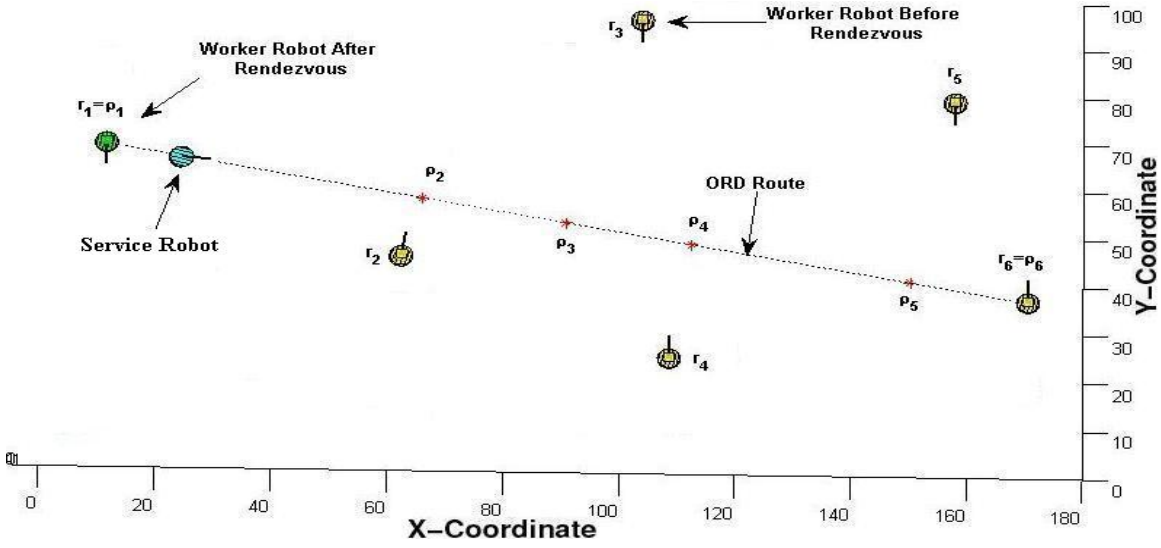

Figure 6.3: Multi-robot, multi-location rendezvous scenario in an obstacle-free environment. The ORD route that passes through the location of the virtual goal $\rho_i \in \mathscr{VG}$ is shown. r_1 and r_6 are the coincidental robots where $\rho_i = r_i$.

route (i.e., the dashed-line in this figure). Furthermore, the cumulative sum of the cost of the relocation of robots of these sets are equal (see equation 2.17 and equation 2.19). The locations of r_1 and r_2 are along the ORD route and coincide with their respective virtual goals (see Chapter 2, Lemma 5 and Lemma 3).

6.3.1 Decision Engine of the Worker Robots

Figure 6.4 and Figure 6.5 show the performance of the decision engine of worker robots in an obstacle-free environment. These figures show the votes of worker robots $r_i, i = 1 \ldots 6$, for the set of virtual goals $\mathscr{VG} = \{\rho_1 \ldots \rho_6\}$, respectively.

The absence of the obstacles in the environment allows the agents to follow their navigational paths to their respective virtual goals without a requirement of the collision avoidance. As a result, these paths are not modified during the rendezvous mission. This has a direct influence on the performance of the decision engine of worker robots. This is due to the fact that the default ranking module of the external state component of the decision engine ranks the virtual goals based on the distances of the robotic agent to the locations of these virtual goals (see section 3.2, Definition 6). The unaltered estimates of the votes by the default ranking module results in the modification of the opportunistic ranking module

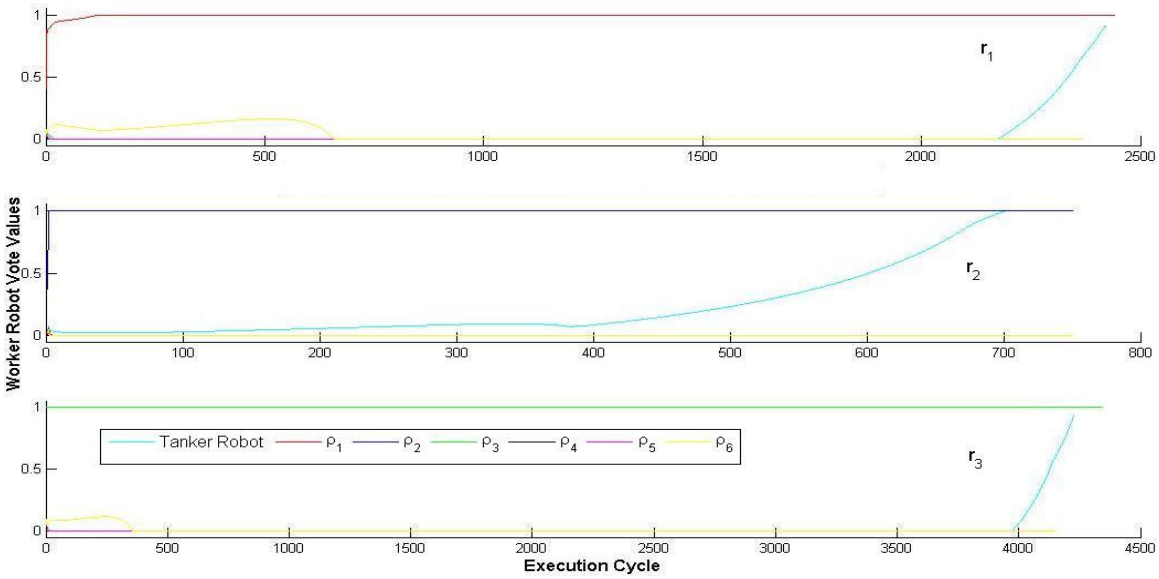

Figure 6.4: The evolution of the votes of the worker robots r_1, r_2, and r_3 for the set of virtual goals $\mathscr{VG} = \{\rho_1 \ldots \rho_6\}$ in an obstacle-free environment.

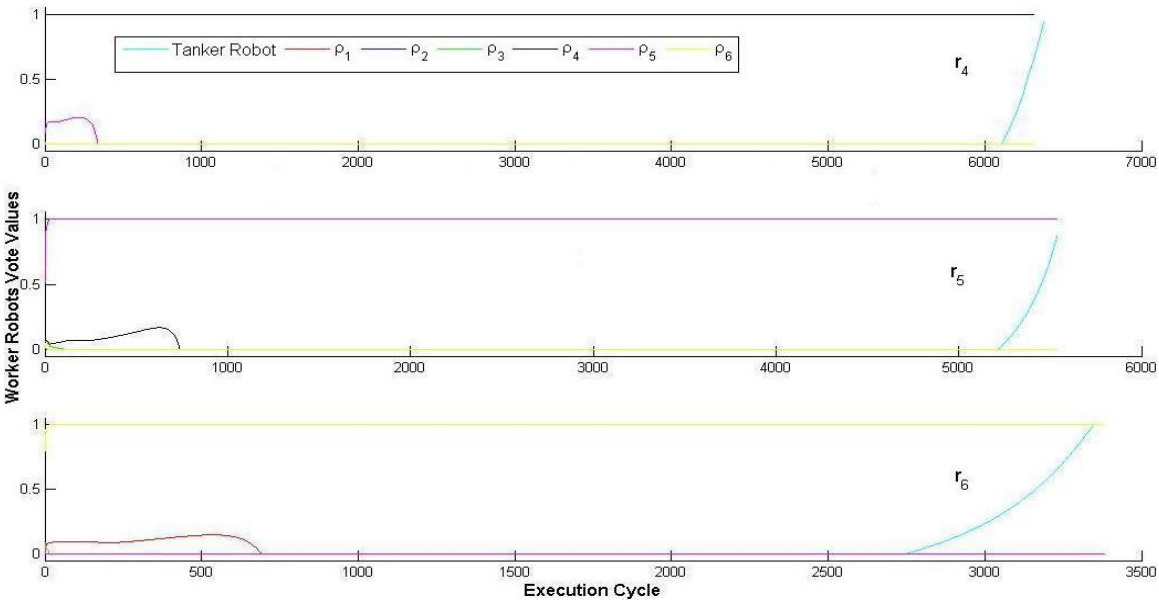

Figure 6.5: The evolution of the votes of the worker robots r_4, r_5, and r_6 for the set of virtual goals $\mathscr{VG} = \{\rho_1 \dots \rho_6\}$ in an obstacle-free environment.

of the external state component of the decision engine to reflect a steady growth of the best choices of the worker robots (see section 3.2, Definition 7 and Figure 3.1). Figures 6.4 and 6.5 show that the best choices of the virtual goals of worker robots (see Chapter 3, Definition 8) remain intact throughout the rendezvous mission. Additionally, these figures depict the convergence of the best choices for virtual goals with the location of the service robot (i.e., the cyan-colored curve in these figures) as the mission progresses. More specifically, these best choices result in optimum relocations to rendezvous with the service robot (see Chapter 3, Theorem 4).

The absence of obstacles in the environment also influences the coordination process of these agents. In particular, coordination emerges without the manipulation of the votes to achieve an optimum allocation of the virtual goals. This emerging coordination is evident in Figure 6.4 and Figure 6.5. These figures reveal that the best choices of the virtual goals of the worker robots are distinct. Furthermore, they ascertain their best choices for virtual goals through a process that is linear in the cardinality of the set of virtual goals $|\mathscr{VG}|$ (see Chapter 3, Lemma 6). It is apparent that the profile matrix of the votes of worker robots is diagonal in this scenario (see Chapter 4, Definition 9). This one-to-one correspondence between the worker robots and their best choices of virtual goals results in the coordination

process where the constraint of the virtual goals is set to one. (see Chapter 4, Definition 10 and Lemma 9).

6.4 Rendezvous in the Presence of Obstacles

Figure 6.6 shows a snapshot of the simulation where a number of stationary rectangular obstacles are introduced to the environment. The presence of the obstacles has a substantial influence on the decision-making and coordination processes. In particular, the result of the votes do not imply an one-to-one correspondence between the worker robots and their virtual goals (see section 6.3.1, Figure 6.1, and Figure 6.2). This is due to the contingency of the change of navigational paths to avoid collision with the obstacles or the other agents in their vicinity. This change of direction effects the choices of the virtual goals of the worker robots. Specifically, the change in direction makes it possible for an agent to be closer to a virtual goal that is different from the elected best choice of virtual in previous decision cycle. Therefore, it is crucial for the system to pay special attention to the constraints of virtual goals during the coordination process (see Chapter 4, Definition 10).

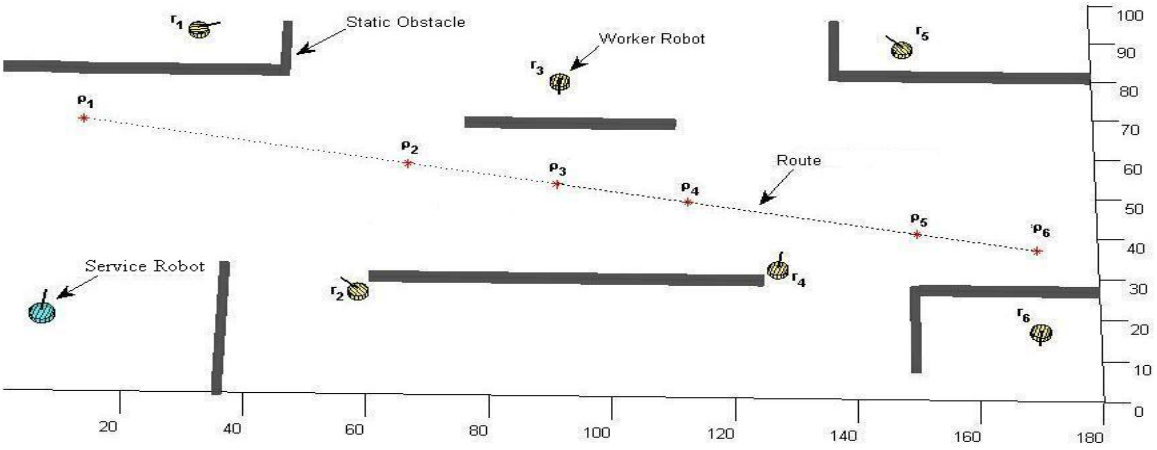

Figure 6.6: Rendezvous in the presence of obstacles. Worker robots are presented in yellow. The service robot is colored in cyan. The rendezvous locations are the red-colored asterisks. A route connecting these virtual goals is presented. Static obstacles are depicted in black.

6.4.1 Agents Votes Maximization Strategy

In this section we study the performance of the agents votes maximization coordination strategy (see section 4.1 and section 4.1.1). We study the performance of this strategy under the effect of the constraints imposed on the virtual goals (see Chapter 4, Definition 10). We consider two generic examples of the constraints of the virtual goals.[2] The setting $q = 0$ exemplifies a scenario where the system applies no restriction on the number of worker robots per virtual goal. On the other hand, the second setting is representative of a situation where the system imposes an upper bound on the number of worker robots that are allowed to attend a given virtual goal. We use the upper bound of two worker robots per virtual goal $q \leq 2$.

The constraint $q = 0$

This constraint of the virtual goals applies no restriction on the number of worker robots that attend a given virtual goal to rendezvous with the service robot. As a result, the worker robots are permitted to meet the service robot at a virtual goal that is elected as its best choice by the decision engine of the agent (see Chapter 3, Definition 8). This indicates that the result of the agents votes maximization strategy to allocate the virtual goals and the best choices of the worker robots are the same (see Chapter 4, Algorithm 4 and section 4.1.1). When $q = 0$, the process of allocation of the virtual goals is reduced to the linear search through the corresponding entries of the worker robots in the profile matrix (see Chapter 4, Definition 9 and Lemma 7).

Figures 6.7 and 6.8 show the votes of the worker robots for the set of virtual goals $\mathscr{VG} = \{\rho_1 \dots \rho_6\}$ when $q = 0$. These figures demonstrate that the decision engine is able to select the best choice of the virtual goal for the individual agents (see Chapter 3, Theorem 4). Figure 6.7 indicates that some worker robots do not detect obstacles along their paths to their respective virtual goals. This is evident in the votes of r_1 and r_3 in Figure 6.7. The change of the best choices of the virtual goals of r_2, r_4, r_5, and r_6 indicates the occurrences of the obstacles along the paths of these robots. Furthermore, Figures 6.7 and 6.8 show that the change of the best choices of the virtual goals is in accordance with the relocation of the service robot within the field. This is verified by the convergence of the curve of the best choice of the virtual goal of the individual worker robots with the vote of the agents

[2]Section 6.3.1 present the case where $q = 1$.

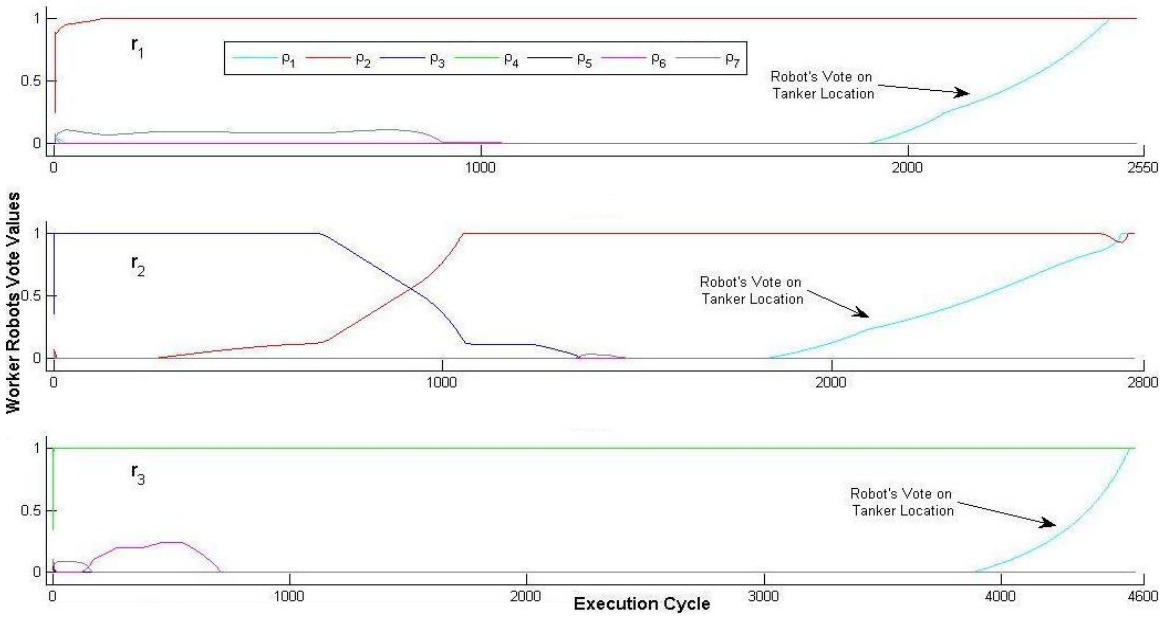

Figure 6.7: The evolution of the votes of the worker robots r_1, r_2, and r_3. Cyan-colored curve corresponds to the location of service robot in every execution cycle.

on the location of the service robot (i.e., cyan-colored curve in these figures).

These figures also represent the allocation of the virtual goals performed by the agents votes maximization strategy. The constraint $q = 0$ enables this strategy to allocate the virtual goals where no restriction on the number of agents per virtual goal is considered. Hence, the final assignments of the agents votes maximization strategy is essentially the best choices of the virtual goals of the worker robots in their respective vote profiles (see Chapter 3, Definition 5).

The constraint $q \leq 2$

We use the same configuration of the service and the worker robots in $q = 0$ to study the performance of the agents votes maximization strategy with the constraint $q \leq 2$. As a result, Figures 6.7 and 6.8 also represent the votes of worker robots in this setting. However, the modification on the constraint of the virtual goals influences the allocation process of the agents votes maximization strategy. This strategy coordinates the allocation of the worker robots to the virtual goals to comply with their constraint when $q \geq 1$. For instance, r_1, r_2, r_4, and r_6 elect the virtual goal $\rho_2 \in \mathscr{VG}$ to rendezvous with the service robot when $q = 0$.

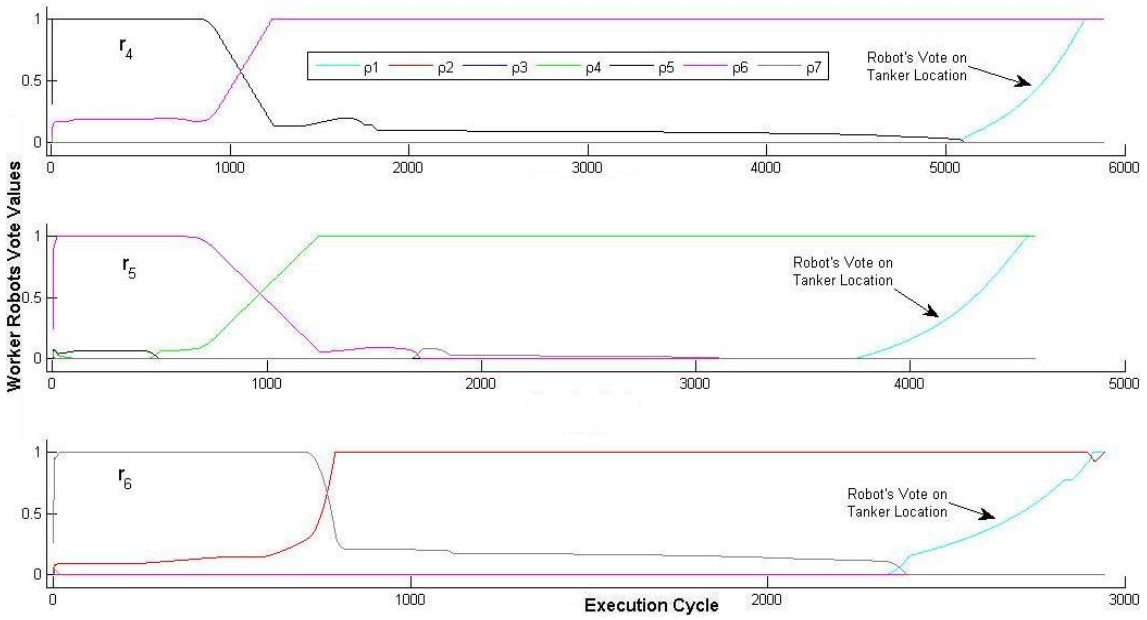

Figure 6.8: The evolution of the votes of the worker robots r_4, r_5, and r_6. Cyan-colored curve corresponds to the location of service robot in every execution cycle.

Furthermore, the agents votes maximization strategy does not modify these choices since they do not violate the constraint of the virtual goals. However, these allocations require further modification to assign two worker robots at most to every virtual goal when $q \leq 2$.

Figures 6.9 and 6.10 present the allocation of the virtual goals to the worker robots where $q \leq 2$. These figures verify that the maximum number of worker robots allocated to a specific virtual goal is in accordance with this constraint. For example, the virtual goals ρ_2 and ρ_4 are allocated to two robots each. These are r_1 and r_3 to ρ_2, and r_4 and r_5 to ρ_4. In addition, the virtual goals ρ_3 and ρ_5 are allocated to one worker robot each. They are r_2 and r_6, respectively. However, the virtual goal ρ_6 is not allocated to any worker robot. An unallocated virtual goal is a verification of the effectiveness of the decision engine in the presence of the obstacles. More specifically, the decision engine is capable of determining the best choices of the virtual goals of the agents (see Chapter 3, Theorem 5). Moreover, this result is unaffected by constraint of the virtual goals.

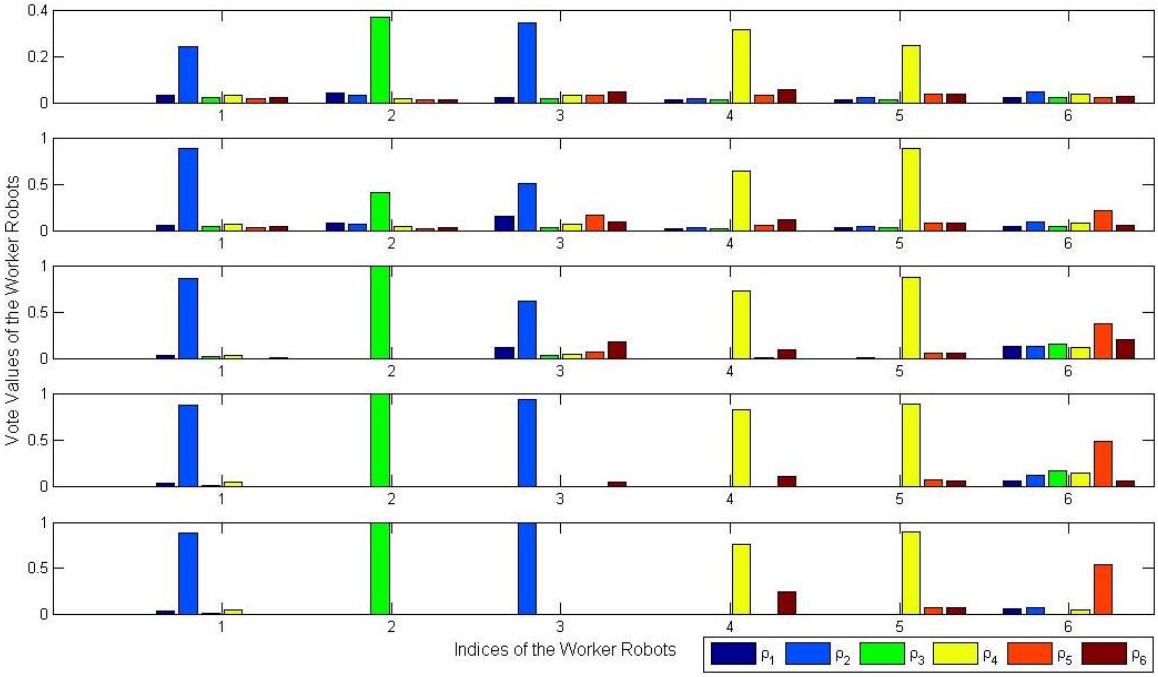

Figure 6.9: The evolution of the allocation of the virtual goals of worker robots using agents votes maximization strategy between 0 to 3000 execution cycles.

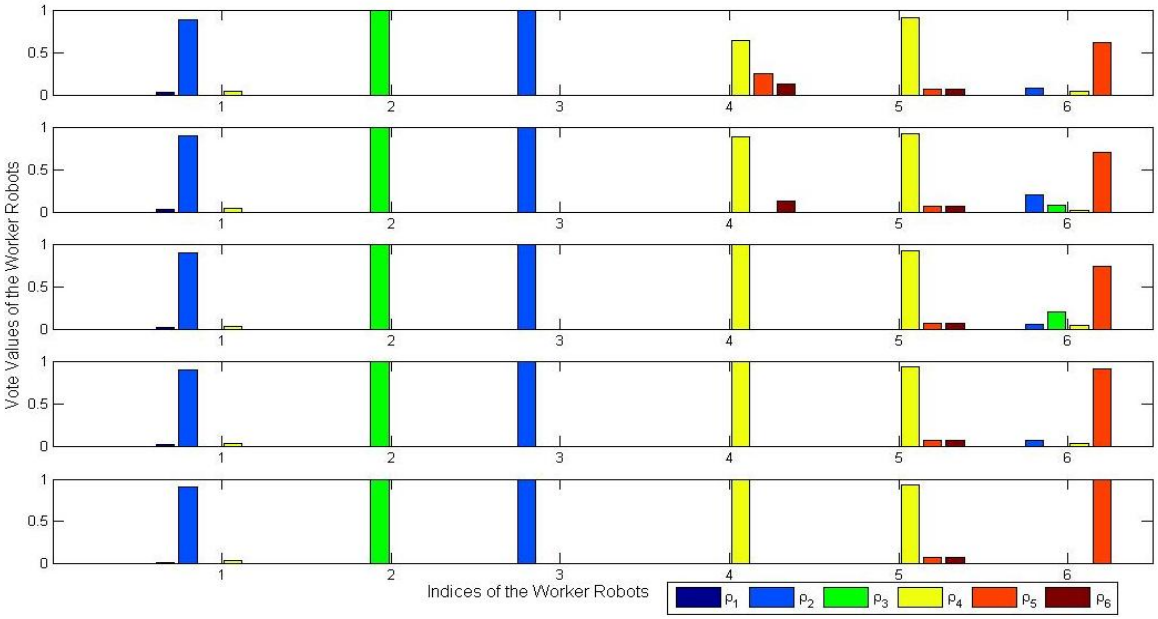

Figure 6.10: The evolution of the allocation of the virtual goals of worker robots using agents votes maximization strategy between 3000 to 6000 execution cycles.

6.5 Further Analysis

We compare the performance of the agents votes maximization strategy with the fixed service station (see Silverman et al., 2002; Oh and Zelinsky, 2000) and a single dynamic rendezvous location strategy (e.g., Zebrowski et al., 2007).[3] We refer to this single dynamic rendezvous location as CoM hereafter.

6.5.1 Obstacle-Free Environment

Figures 6.11 and 6.12 show the travel distance of the worker robots and the cumulative sum of the travel distance of the worker and the service robots for rendezvous, respectively. The y-coordinates of these figures are numbered 1 through 4 to indicate the different distributions of the worker robots and the displacements of the service robot during the experiments (see section 6.1, case 4 and case 5). Additionally, these figures show the result of the travel distance of the robotic agents where different cost of the relocation is utilized to calculate the set of virtual goals using the LSRD decomposition (see Appendix A, and Figure 6.2). The costs of the relocation of the worker robots are $w_i = \frac{1}{\sigma_i^2}$, $w_i = \frac{1}{\sqrt{x_i}}$, $w_i = \frac{y_i}{x_i}$, $w_i = \frac{1}{y_i}$, and $w_i = 1$. The top most bar in every y-coordinate entry of these figures represents the travel distance of the worker robots to a fixed service station.[4] This fixed station is relocated in the environment using the same setting described for the service robot during the experiments (see section 6.1, case 5). The results of the travel distance of the robotic agents using the ORD decomposition is the second bar from the top in every y-coordinate entry.

Figure 6.11 and Figure 6.12 indicate that the travel distance of the robotic agents using the LSRD decomposition outperforms the fixed station strategy. Moreover, this difference is intact by the modification of the cost of the relocation of the worker robots. A comparison between Figure 6.11 and Figure 6.12 reveals that the LSRD decomposition retains this improvement over the fixed station strategy when the cumulative sum of the travel distance of the service and worker robots is considered. However, the ORD decomposition (see Chapter 2, Algorithm 3) enables the robotic agents to accomplish the rendezvous mission with a significantly shorter travel distance compared to the fixed station and the LSRD

[3]Fermat point, Fermat-Toricelli point, Weber point and center of the mass are used interchangeably to refer to this location.

[4]The fixed station entry in Figure 6.12 is the replicate of the results of this strategy in Figure 6.11. It is provided in Figure 6.12 for the simplicity of the comparison of the results of the different strategies when the cumulative sum of the travel distance of the service and the worker robots is considered.

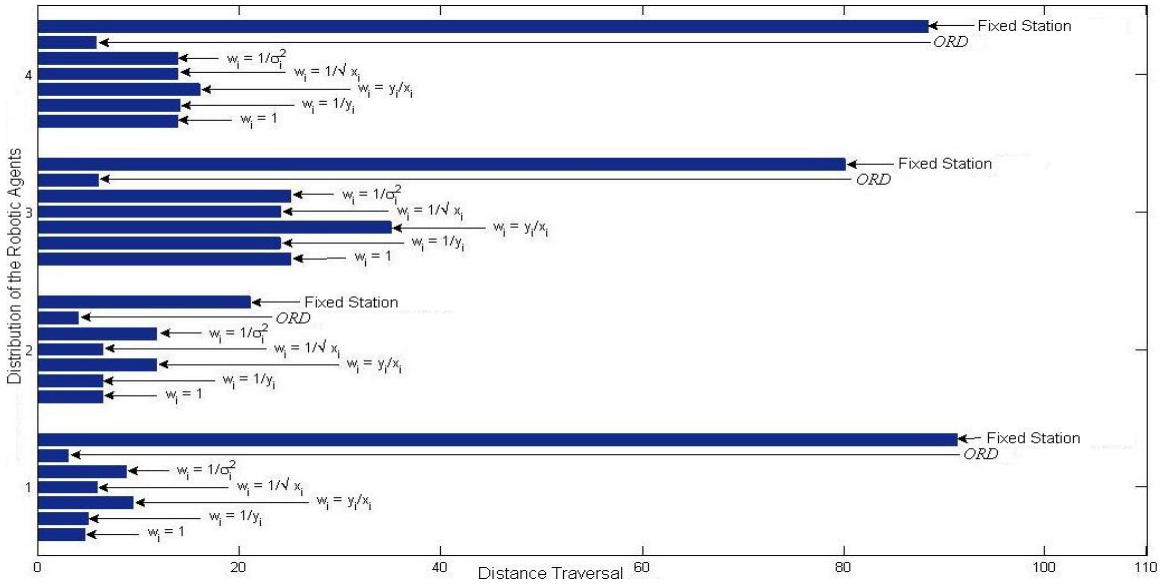

Figure 6.11: The distance traveled by the worker robots. The worker robots are located (1) closest to each other and farthest from the service robot, (2) as close as possible to the location of the service robot, (3) arbitrarily in the environment, and (4) to the farthest distance from the location of the service robot.

decomposition strategies. Figure 6.11 shows the significant difference in the travel distance using the ORD decomposition. Furthermore, this figure reveals that the travel distance using the ORD decomposition is not highly affected by the distributions of the worker robots. A comparison between the y-coordinate entries of Figure 6.11 verifies that differences in the travel distance of the worker robots using the ORD decomposition is negligible. These results demonstrate the invariance of the ORD decomposition to the distributions of the location information of robotic agents (see section 2.2.1). However, the initial location of the service robot has a considerable impact on the cumulative sum of the travel distance of the robotic agents using the ORD decomposition. The influence of the initial location of the service robot in conjunction with the distributions of the worker robots is verified through the comparison of the y-coordinate entries of Figures 6.11 and 6.12.

Table 6.1 summarizes the means, the medians, and the standard deviations of the travel distances of the service and the worker robots using the ORD, the LSRD, the fixed station and the CoM strategies. This table shows that the travel distance of the service robot is unaffected by the LSRD and the ORD decomposition strategies. However, the service robot achieves a better travel distance using these strategies compared to the CoM approach.

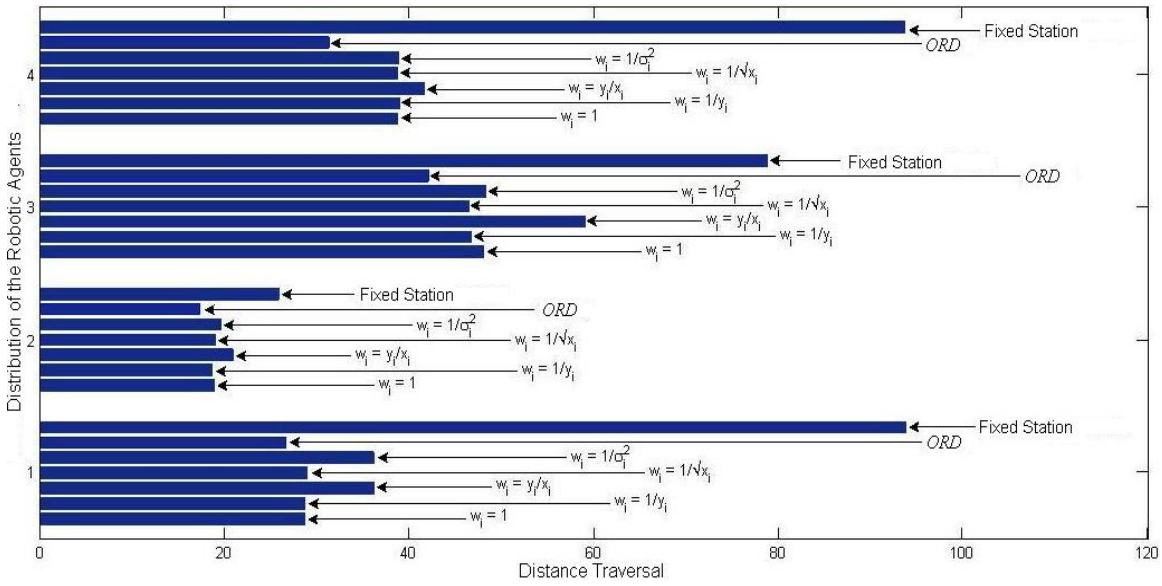

Figure 6.12: The cumulative sum of the distance traveled by the service and worker robots. The worker robots are located (1) closest to each other and farthest from the service robot, (2) as close as possible to the location of the service robot, (3) arbitrarily in the environment, and (4) to the farthest distance from the location of the service robot.

Additionally, the standard deviation of the travel distance of the service robot supports the improvement of the mean travel distance using the ORD and the LSRD decompositions. Table 6.1 indicates that the standard deviation of the travel distance of the service robot using the CoM is not within one standard deviation of the ORD and the LSRD. In addition, this observation is unaffected by the different costs of the relocation of worker robots. On the other hand, the CoM strategy outperforms the fixed station strategy.

In contrast, the impact of the ORD and the LSRD strategies on the travel distance of the worker robots is significant. Table 6.1 indicates that the ORD decomposition yields an optimal relocation strategy that minimizes the cumulative sum of the travel distance of the robotic agents in a rendezvous mission (see Chapter 2, Lemma 4 and Theorem 1). In particular, the ORD decomposition enables the robotic agents to achieve a rendezvous strategy where the cumulative sum of the travel distance of the service and the worker robots outperforms the result of the CoM strategy (see Chapter 2, Corollary 1).

It is also apparent that the overall performance of the LSRD decomposition is better than the CoM and the fixed station strategies. However, the effect of the different costs of

Table 6.1: The mean, the median, and the standard deviation of the travel distance of the robotic agents using the ORD, the LSRD, the CoM and the fixed station strategies in an obstacle-free environment.

Recharging	Worker Robots			Service Robot		
Strategies	Mean	Median	STD	Mean	Median	STD
ORD	4.09	1.26	5.00	20.21	3.50	20.00
$w_i = 1$	9.88	2.59	9.00	20.35	3.88	20.00
$w_i = \frac{1}{\sigma_i^2}$	10.95	2.98	10.90	20.98	3.48	20.44
$w_i = \frac{1}{\sqrt{x_i}}$	9.77	2.62	9.00	20.07	3.72	20.00
$w_i = \frac{1}{y_i}$	10.01	2.71	9.05	20.00	3.67	20.00
$w_i = \frac{y_i}{x_i}$	15.51	4.11	13.00	20.01	3.68	20.00
CoM	43.67	13.28	26.89	32.16	5.43	23.53
Fixed Station	65.54	16.80	58.06	-	-	-

relocation of the worker robots on the performance of the LSRD decomposition is insignificant. Table 6.1 shows that the results of the travel distance of the robotic agents using different cost of relocation are within one standard deviation of each other. Thus, drawing conclusions on the effect of the cost of relocation on the performance of the LSRD decomposition is not warranted. The only exception is with regards to the cost of the relocation $w_i = \frac{y_i}{x_i}$. However, this weighting criterion attains the highest mean of the travel distance when the LSRD decomposition is used.

6.5.2 Presence of the Obstacles

Table 6.2 shows the means, the medians, and the standard deviations of the travel distances of the service and the worker robots using the ORD, the LSRD, the fixed station, and the CoM strategies in the presence of obstacles. The entries of this table verify that the CoM strategy achieves a shorter travel distance compared to the fixed station. This observation holds when $w_i = \frac{y_i}{x_i}$ is used to calculate the cost of the relocation of the worker robots using the LSRD decomposition. Furthermore, the ORD decomposition outperforms these strategies. A comparison between Tables 6.2 and 6.1 reveals a change in performance of the LSRD decomposition in conjunction with specific choices of the cost for the relocation of the worker robots. More specifically, the standard deviation of the LSRD decomposition with $w_i = 1$ and $w_i = \frac{1}{\sqrt{x_i}}$ are within one standard deviation of the travel distance of the

Table 6.2: The mean, the median, and the standard deviation of the travel distance of the robotic agents using the ORD, the LSRD, the CoM, and the fixed station strategies in the presence of obstacles.

Recharging Strategies	Worker Robots			Service Robot		
	Mean	Median	STD	Mean	Median	STD
ORD	49.29	38.56	39.67	73.10	61.44	47.47
$w_i = 1$	68.64	67.52	38.83	88.40	77.52	57.10
$w_i = \frac{1}{\sigma_i^2}$	68.68	67.24	36.13	88.72	77.12	57.77
$w_i = \frac{1}{\sqrt{x_i}}$	69.35	66.08	39.02	91.09	76.00	62.00
$w_i = \frac{y_i}{\sqrt{x_i}}$	83.15	78.08	32.28	88.93	75.83	67.19
CoM	82.32	85.36	22.10	58.82	10.36	39.10
Fixed Station	164.93	132.88	48.38	-	-	-

ORD decomposition. Hence, the difference between the mean travel distance of the LSRD and the ORD decompositions are not warranted when the costs of the relocation are used. However, the result of the ORD decomposition outperforms the LSRD strategy when the travel distance of the service robot is considered. Furthermore, this result is unaffected by the weighting criteria to relocate the worker robots. Table 6.2 verifies that the presence of obstacles in the environment does not influence the LSRD when the different costs of the relocation of worker robots are compared.

Table 6.3 shows the effect of the opportunistic ranking module of the external state component of the decision engine (see section 3.2).[5] Specifically, this table demonstrates the effect of this module on the travel distance of the ORD decomposition in comparison to the CoM and the fixed station strategies in the presence of obstacles. It is apparent that the ORD decomposition exhibits a significant improvement in the travel distance compared to the CoM and the fixed station strategies. Moreover, this improvement is achieved in the travel distance of the worker robots as well as the cumulative sum of the distances traveled by the service and the worker robots. This verifies the theoretical aspects of the ORD decomposition presented in Chapter 2 (see Lemma 4, Theorem 1, and the Corollary 1).

Table 6.3 indicates that the improvement of the cumulative sum of the travel distance of the service and worker robots is negligible when the opportunistic ranking module is used.

[5]The entry *System* refers to the cumulative sum of the travel distance of the service and worker robots.

Table 6.3: The effect of the opportunistic ranking module of the external state component of the decision engine on the mean and standard deviation of the travel distance.

	Mean		STD	
	Workers	System	Workers	System
ORD ($\omega_i(\rho_j)$ present)	49.29	122.39	39.35	87.82
ORD ($\omega_i(\rho_j)$ absent)	56.67	133.05	45.98	86.57
CoM	82.32	140.14	22.10	61.20
Fixed Station	164.93	–	48.38	–

This module improves the travel distance of the worker robots. However, the standard deviation of the cumulative sum of the travel distance at the group-level in the presence and the absence of this module are within one standard deviation and are statistically insignificant.

6.6 Discussion

Every mechanism, capable of interacting with its environment, biological or otherwise, requires energy for survival. Robotic agents are no exception. It is vital for these autonomous mobile mechanisms, individually or as a unit, to provide themselves with the opportunity to gather energy.

The multi-robot recharging problem is a scenario where the results of this research are applicable. The multi-location rendezvous problem is analogous to the problem of the computation of a number of recharging locations in the field of the operation of the robotic agents. Furthermore, it is important to select these rendezvous locations to minimize the amount of the energy that robots expend on the recharging process.

In this chapter, we demonstrated how distributional information of the robotic agents enables a multi-robot system to determine the ideal rendezvous locations. In addition, we achieved this rendezvous mission without the requirement of a predefined fixed station. In a recharging mission, the multi-location rendezvous decomposition along with the elimination of the requirement of a fixed station reduce navigational efforts of the individual robots to the station and their work places. Moreover, we demonstrated that the combination of linear decomposition and the agents votes maximization allocation strategy yield the minimization of the travel distance of the robotic agents to their respective rendezvous locations.

This increases the energy preservation of the entire system in a recharging operation. Additionally, we verified that the performance of this strategy is unaffected by the situation where the rendezvous locations are constrained by the number of robotic agents.

It is possible to extend the results of this study. For instance, it is possible to enhance the cost of the relocation to address the uncertainties that exist in the location information of the robotic agents. Alternatively, these costs can be formulated to estimate the contingent correlation between the rendezvous time and the energy level or the dynamics of the robotic agents. The effect of the environmental condition such as the type of the terrain can be formulated through the costs of the relocation of the agents.

Furthermore, this research can be extended to analyze the behavior of the system through the introduction of multiple service robots. This problem require another layer of the coordination that cooperatively distributes the worker robots among the service robots to rendezvous locations.

The type of the service provided to a multi-robot system has a direct impact on the performance of the adapted strategy. This emphasizes the necessity to analyze the results of this research under specific problem characterizations and specifications. For example, it is possible to study the starvation state of the entire system and the energy limit of the service robot in a multi-robot, multi-location recharging scenario.

Figure 6.13: Multi-robot, multi-location rendezvous in the presence of obstacles.

Chapter 7

Multi-Robot Single-Intruder Pursuit

Pursuit and evasion has received special attention in research of multi-robot systems. They offer a wide range of real-life applications from gaming and the military to border patrol and other domains. Jankovic (1978) proves that in a polygonal environment three pursuers are sufficient to capture an intruder if the initial location of the intruder is within the convex hull of the locations of the pursuers.[1] We extend this result to show that the confinement of the initial location of the intruder within the convex of the locations of pursuers is not a necessary condition. We demonstrate that this confinement is alleviated through the incremental computation of a set of virtual goals that are independent from the locations of the pursuers. We consider the location information of the intruder to present the isogonic point of an isosceles triangle to generate this set of virtual goals (see equation 2.39 through equation 2.42 and equation 2.50). These virtual goals form the vertices of this triangle (see Figure 2.7). We use variable lengths of the side and the height of this conceptual triangle to provide flexibility in the positioning of these virtual goals. This set of virtual goals reflects the relocation of the intruder at every execution cycle (see equation 2.52 and equation 2.53). This results in the location of the intruder within the convex of these virtual goals (see Chapter 2, Theorem 3 and Corollary 3).

[1] Kopparty and Ravishankar (2005) generalize this result and prove that the number of pursuers is proportional to the dimension of the environment. They show that in a $\mathbb{R}^d$, $d \geq 2$ polygonal environment, $d+1$ pursuers achieve the same result. However, this generalization is beyond the scope of the present case study.

We study the effect of the opportunistic ranking module of the external state component of the decision engine of pursuers on the decision process (see Chapter 3, Definition 7 and equation 3.7). Furthermore, we study the performance of the agents votes maximization and the profile matrix permutations strategies to coordinate the robotic agents (see section 4.1, Algorithm 4 and section 4.2, Algorithm 5). We consider the time, the energy expended, and the distance traveled by the pursuers as metrics to analyze the performance of these strategies in contrast to three different allocation strategies. They are the probabilistic (see Appendix D), the leader-follower (e.g., Undeger and Polat, 2010), and the prioritization coordination strategies.[2]

The remainder of this chapter is organized as follows. Section 7.1 describes the simulation setup. Section 7.2 presents the process of the generation of a set of virtual goals using the location information of the intruder. In addition, this section illustrates the performance of the decision engine of the pursuers to rank these virtual goals. Moreover, the process of the virtual goals allocation to the pursuers using the agents votes maximization and the profile matrix permutations strategies is presented. We analyze the performance of these coordination strategies in section 7.3. We conclude this chapter in section 7.4.

7.1 Simulation Setup

The simulation environment consists of a number of stationary rectangular obstacles. There are two exits in this environment. We consider a multi-robot, single intruder pursuit scenario. There are three pursuers involved in every experiment. The linear velocity of the intruder and the pursuers varies between 0 and $10\frac{m}{s}$. In addition, these robotic agents interact with the environment using their respective on-board simulated sensors. They perform simple reactive collision avoidance to avoid collision with the obstacles and the robotic agents in their vicinity.

1. **Task of the intruder** : The intruder enters this environment from one of the exits and trespasses to the other exit. This evasive agent attempts to escape if it detects a pursuer. However, the intruder does not follow any specific escape plan. It changes its navigational direction until the pursuers are out of its sensing range. We choose

[2]Leader-follower strategy designates one of the robots as the leader of a multi-robot system. These robots follow the strategy that is formulated using the information of the leader (e.g., location information). The prioritization implies a deterministic assignments of tasks at the commencement of a mission

the initial location of the intruder as follows (see Figure 7.1).

- The intruder enters the environment from *Exit 1* and trespasses to *Exit 2*.
- The intruder enters the environment from *Exit 2* and trespasses to *Exit 1*.
- The initial location of the intruder is selected arbitrarily in the environment. The intruder chooses the closest exit and moves towards this exit.

2. **Location of the pursuers** : We locate the pursuers in the bottom-left corner compound of the simulation environment (referred to as the base of the pursuers in Figure 7.1). We use the same initial location information of the pursuers in all the experiments.

3. **Experiments** : We use the initial locations of the intruder to determine the effect of the placement of the intruder on the pursuit missions. A pursuit mission is successful if the intruder falls within the convex hull of the locations of the pursuers such that any further movement of the intruder results in collisions with the pursuers.[3]

7.2 Isogonic Decomposition Pursuit

Figure 7.1 shows a snapshot of the simulation environment. The pursuers are shown in green. The red-colored agent is the intruder. The two exits are labeled. The black-colored rectangles are the stationary obstacles. The base of the pursuers is shown in the bottom-left corner of this figure. A pursuit mission consists of two phases: The ambush phase and the capture phase.

7.2.1 Ambush Phase

The ambush phase instructs the pursuers to leave their base compound to spread out in the environment. This is achieved through the application of a set of virtual goals that are calculated using the location information of the two exit ways and the pursuers. We use the location information of the pursuers to calculate the center of mass of the initial locations

[3]This description of a successful pursuit is highly domain-specific. For instance, an intruder surrenders if it is detected (i.e., tagged) by a pursuer in visibility-based approaches (see Thunberg and Orgen (2010) for example).

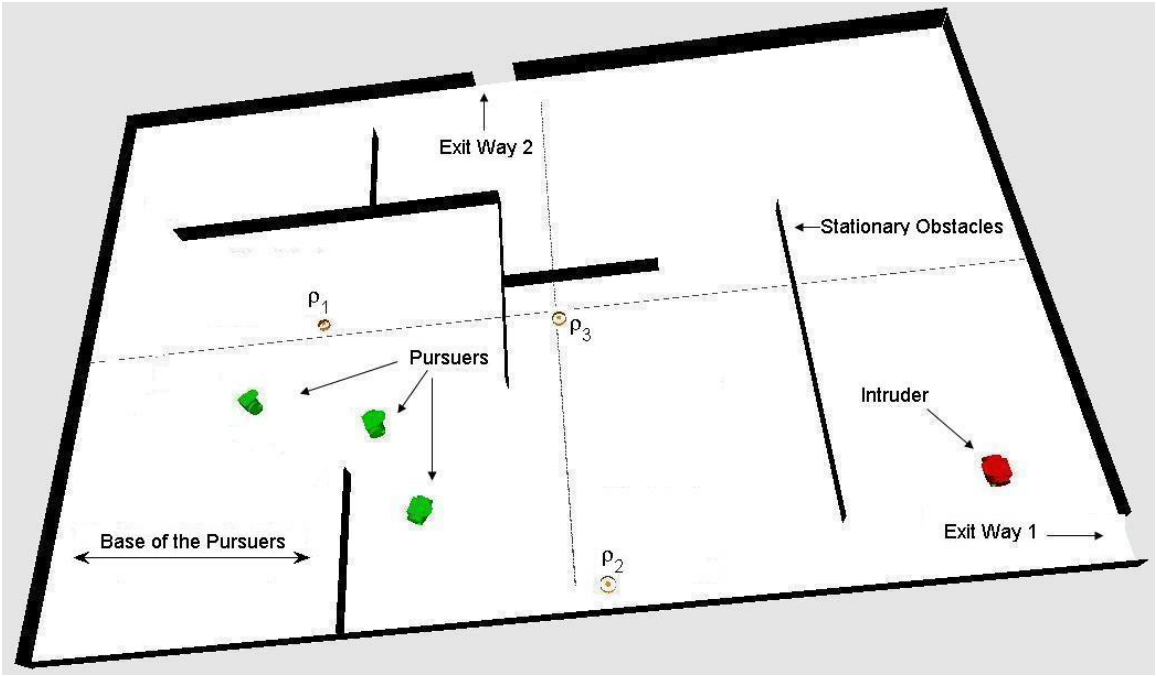

Figure 7.1: Multi-robot, single intruder pursuit. The intruder is the red-colored agent. The pursers are depicted in green. The stationary obstacles are shown in black. The environment consists of two exits. The virtual goals are shown by the orange-colored circles.

of these agents as:

$$\mathscr{P}_r = argmin_p \sum_{i=1}^{3} \|r_i - p\| = \frac{1}{3} \sum_{i=1}^{3} r_i \tag{7.1}$$

Where r_i and $\mathscr{P}_r$ denote the location information of the i^{th} robotic agent and the center of mass of the initial locations of the pursuers. We utilize $\mathscr{P}_r$ and the location information

of the two exits to calculate a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ as:

$$\rho_j = \frac{1}{2}\|\mathscr{P}_r - ext_j\|,\ j = 1, 2 \tag{7.2}$$

$$\rho_3 = argmin_p[\sum_{j=1}^{2} \|ext_j - p\| + \|\mathscr{P}_r - p\|]$$

$$= \frac{1}{3}[\sum_{j=1}^{2} ext_j + \mathscr{P}_r] \tag{7.3}$$

where ext_j represents the location information of the j^{th} exit. As a result, the virtual goals ρ_1 and ρ_2 are placed between the base compound and the two exits. In contrast, the virtual goal ρ_3 is placed approximately in the middle of the environment. The virtual goals ρ_1 and ρ_2 relocate the pursuers closer to the exits. The virtual goal ρ_3 relocates one of the pursuer to the middle of the environment to facilitate the entrapment of the intruder during the capture phase (see section 7.2.2).

Decision Engine of the Pursuers during the Ambush Phase

Figure 7.2 shows the evolution of the votes of pursuers r_1, r_2, and r_3 for the set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ calculated in section 7.2.1. Subplots (A), (B), and (C) correspond to the votes that are computed by the default ranking module $\pi_i^t(r_i \mapsto \rho_j)$ of the external state component of the decision engine of the pursuers (see Chapter 3, Definition 6). Subplots (D), (E), and (F) represent the votes that are computed by the decision engine of the agents after the incorporation of their respective opportunistic ranking modules $\omega_i(\rho_j)$ (see Chapter 3, Definition 7).

Subplots (A), (B), and (C) indicate that the decision engine of the pursuers expends a longer time in ranking the available virtual goals. This delay of the ranking of virtual goals is verified through the neutral votes of the pursuers. The default ranking module of the external state component of the pursuers rank the virtual goals with almost the same fixed value 0.33 for more than 400 execution cycles. The neutrality of the votes of the default ranking module changes only after the distances of the pursuers to the virtual goals decrease. However, the maximum value of these votes does not exceed 0.75.[4]

[4]The range of the vote values of the robotic agents is on [0 1] interval such that their normalized cumulative sum equals 1 (see Chapter 3, Definition 5 and equation 3.2).

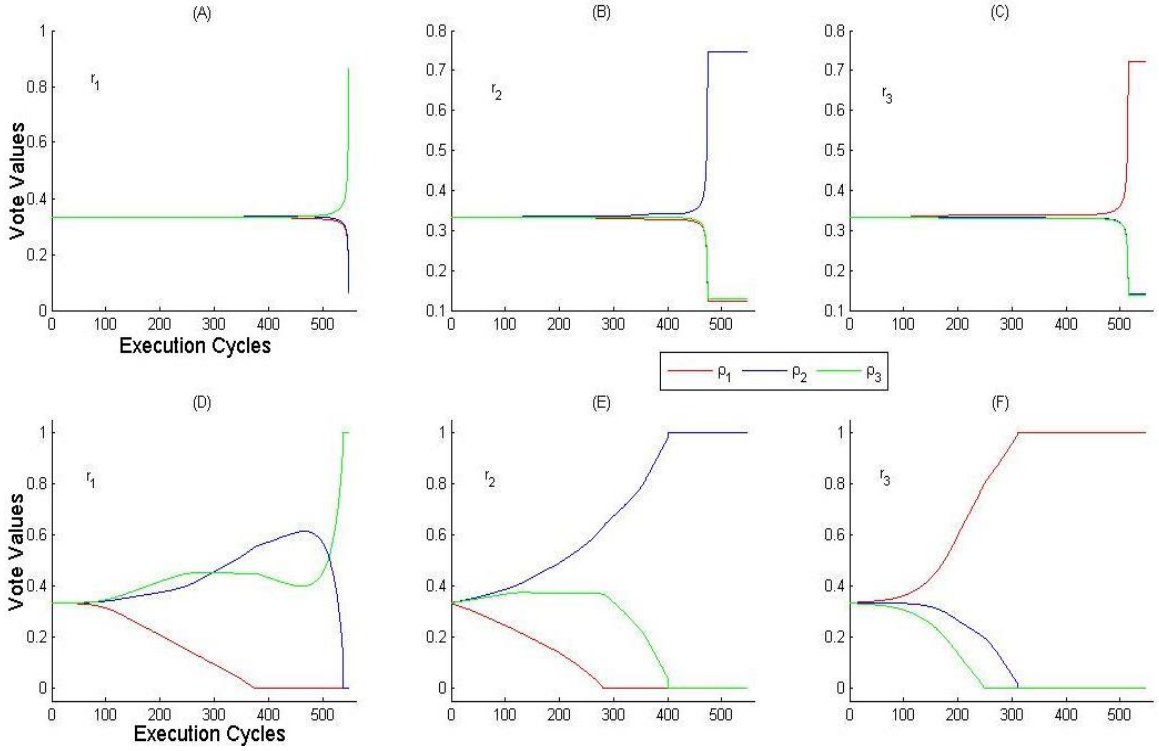

Figure 7.2: The evolution of the votes of pursuers r_1, r_2, and r_3 during the ambush phase of pursuit. Subplots (A), (B), and (C) represent the votes of the default ranking module $\pi_i^t(r_i \mapsto \rho_j)$ of the agents. Subplots (D), (E), and (F) are the votes of the pursuers after the incorporation of the opportunistic ranking module $\omega_i(\rho_j)$ (equation 3.9).

The incorporation of the opportunistic ranking module of the external state component of the decision engine addresses the delays in the evolution of votes. In subplots (D), (E), and (F), it is apparent that votes associated with different virtual goals evolve from the early stages of the ambush phase. In particular, the decision engine of r_2 and r_3 specify the best choices of the virtual goals of these agents before 400 execution cycles (see Chapter 3, Theorem 4). Furthermore, they distinguish their best choices of virtual goals where these goals are equally ranked (see Chapter 3, Theorem 6). However, r_1 takes longer to choose its best virtual goal. Subplot (D) in Figure 7.2 indicates that the decision engine of r_1 favors the virtual goal ρ_2 (i.e., blue-colored curve) with a higher rank after 300 execution cycles. However, this virtual goal is voted on by r_2 with a higher value at this execution cycle. This

causes the coordination strategy to assign the virtual goal ρ_2 to r_2 and redirect the decision engine of r_1 to the unallocated virtual goal ρ_3 (see section 7.2.3 for further details).

Figure 7.3 and Figure 7.4 show the distributions of the votes of the pursuers for the set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ during the ambush phase. They show the frequencies of the votes of the default and the opportunistic ranking modules for the individual virtual goal $\rho_j \in \mathscr{VG}$. Figure 7.3 clarifies the neutrality of the votes of the default ranking module during this phase. The frequency of these votes mostly spread on [0.30 0.40] interval. The exceptions to this observation are the subplots that represent the virtual goals with the highest votes. These are $\pi_1(\rho_2)$, $\pi_2(\rho_3)$, and $\pi_3(\rho_1)$.[5] However, the value of these votes does not exceed 0.75 and is in accordance with the Figure 7.2.

The votes of the pursuers exhibit wider distributions when the opportunistic ranking module is incorporated into the decision engine. Figure 7.4 indicates that the value of these votes covers the [0.0 1.0] interval. The best choices for the virtual goals of r_2 and r_3 are

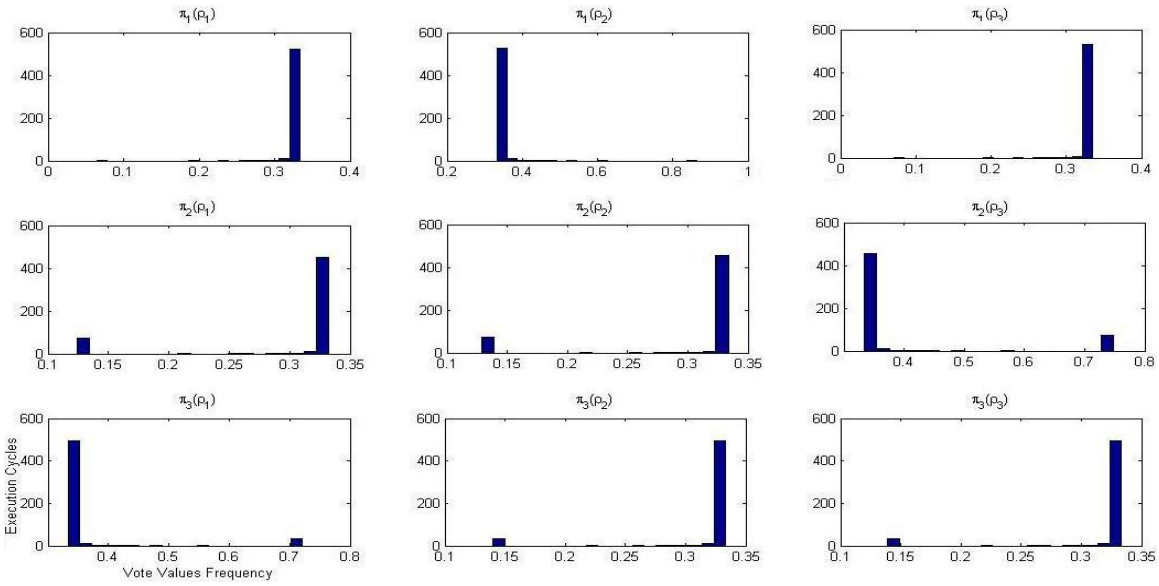

Figure 7.3: Distributions of the votes of default ranking module $\pi_i^t(r_i \mapsto \rho_j)$ of pursuers for a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ during the ambush phase of a pursuit mission. These histograms provide the frequencies of the votes for the virtual goal $\rho_j \in \mathscr{VG}$. $\pi_i(\rho_j)$ represents the vote of the i^{th} pursuer for the j^{th} virtual goal (equation 3.1).

[5] $\pi_i(\rho_j)$ refers to the vote of the i^{th} pursuer for the j^{th} virtual goal (equation 3.1).

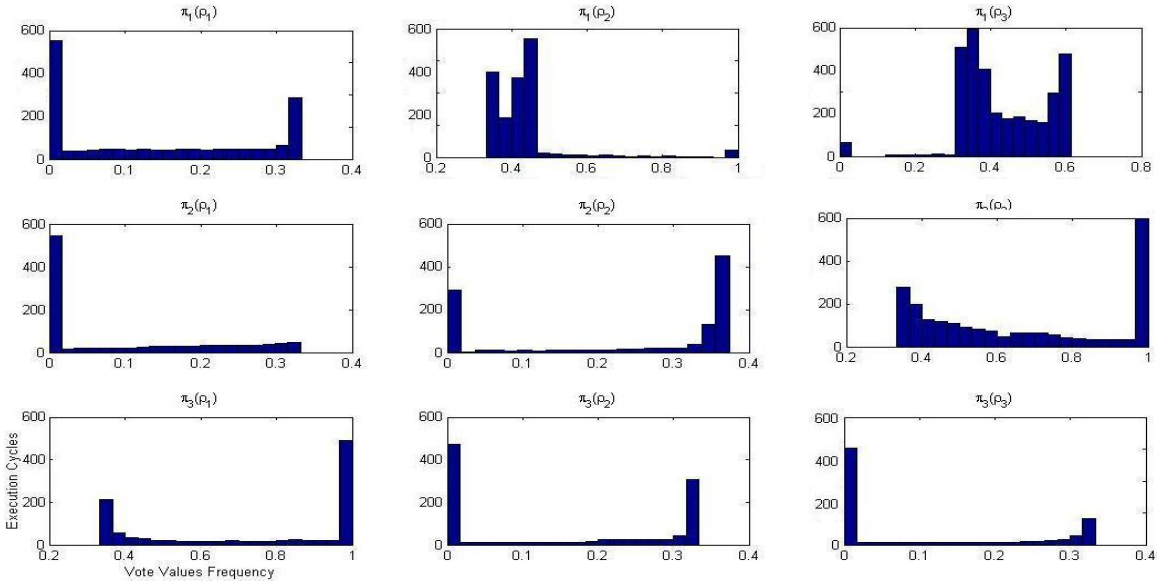

Figure 7.4: Distributions of the vote of pursuers for a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ after the incorporation of the opportunistic ranking module $\omega_i(\rho_j)$ into decision mechanism during the ambush phase of a pursuit mission. These histograms provide the frequencies of the votes for the virtual goal $\rho_j \in \mathscr{VG}$. $\pi_i(\rho_j)$ represents the vote of the i^{th} pursuer for the j^{th} virtual goal (equation 3.1).

apparent in this figure. They are subplots $\pi_2(\rho_3)$ and $\pi_3(\rho_1)$. The best choice for the virtual goal of r_1 shows a shorter bar on the value 1.0 in the subplot $\pi_1(\rho_2)$. An investigation of the subplots of r_1 (i.e., $\pi_1(\rho_1)$, $\pi_1(\rho_2)$, and $\pi_1(\rho_3)$ in Figure 7.4) reveals that the decision engine of this agent favors the virtual goal ρ_3 for an extended period of the ambush phase. This is evident in subplot $\pi_1(\rho_3)$. However, r_2 ranks this virtual goal with a higher vote. As a result, the vote of r_1 exhibits a slower evolution for the virtual goal ρ_2. This situation arises when the decision engine of two or more pursuers favor the same virtual goal as the best choice. We resolve this issue in section 7.2.3.

7.2.2 Capture Phase

Theorem 2 and Corollary 2 in Chapter 2 demonstrate that the displacement of the isogonic point ρ_4 of an isosceles traingle is always in an equal distance from the vertices ρ_2 and ρ_3 (see Figure 2.7). The location of this virtual goal is confined within the convex hull

of the vertices of this triangle if the necessary condition in Theorem 3 is satisfied (see Corollary 3).[6] This reduces the amount of information required to decompose a mission into a set of virtual goals. Specifically, the location information of the intruder suffices to generate the set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$. We denote the location information of the intruder with ρ_4 (i.e., the isogonic point of the triangle $\triangle\rho_1\rho_2\rho_3$ in Figure 2.7). We utilize ρ_4 and equation (2.50) to calculate the location of the virtual goal ρ_1. Subsequently, we use the location information of ρ_1, equation (2.39) and equation (2.40) to calculate the location information of the virtual goals ρ_2 and ρ_3 with regards to the location information of the intruder. In addition, we reflect the relocations of the intruder during the pursuit mission using equation (2.52) and equation (2.53). We use the location information of the intruder (i.e., ρ_4) and equation (2.53) to update the location information of ρ_1 at every execution cycle. Next, we utilize ρ_1 and equation (2.52) to update the location information of the virtual goals ρ_2 and ρ_3.

Decision Engine of the Pursuers during the Capture Phase

Figure 7.5 illustrates the performance of the decision engine of the pursuers during the capture phase. Subplots (A), (B), and (C) correspond to the default ranking module $\pi_i^t(r_i \mapsto \rho_j)$ of the pursuers. The votes of the pursuers after the incorporation of the opportunistic ranking module are presented in subplots (D), (E), and (F). Figure 7.5 signifies the influence of the relocations of the intruder on the decision-making process. The location information of the virtual goals are updated at every execution cycle to reflect the relocation of the intruder. This results in the fluctuation of the rankings of these virtual goals between consecutive execution cycles. A comparison between Figure 7.2 and Figure 7.5 indicates more engagement of the default ranking module in this phase. However, further investigation of Figure 7.5 reveals extreme indecisiveness of this module on the ranking of the virtual goals. The value of these votes are unstable after 900 execution cycles. For example, the maximum vote value achieved by the default ranking module of r_1 is less than 0.36. In addition, the default ranking module of r_3 is indifferent to the available virtual goals for more than 500 execution cycles. Subplot (C) in Figure 7.5 verifies that the values of these votes are fixed at 0.33. Moreover, the votes of r_2 and r_3 alternate frequently between every virtual goals. These votes do not stabilize after 900 execution cycles.

[6] In Figure 2.7, we bound the value of $\angle\lambda$ within the boundary limit $0 \leq \lambda < 120$ to satisfy this condition.

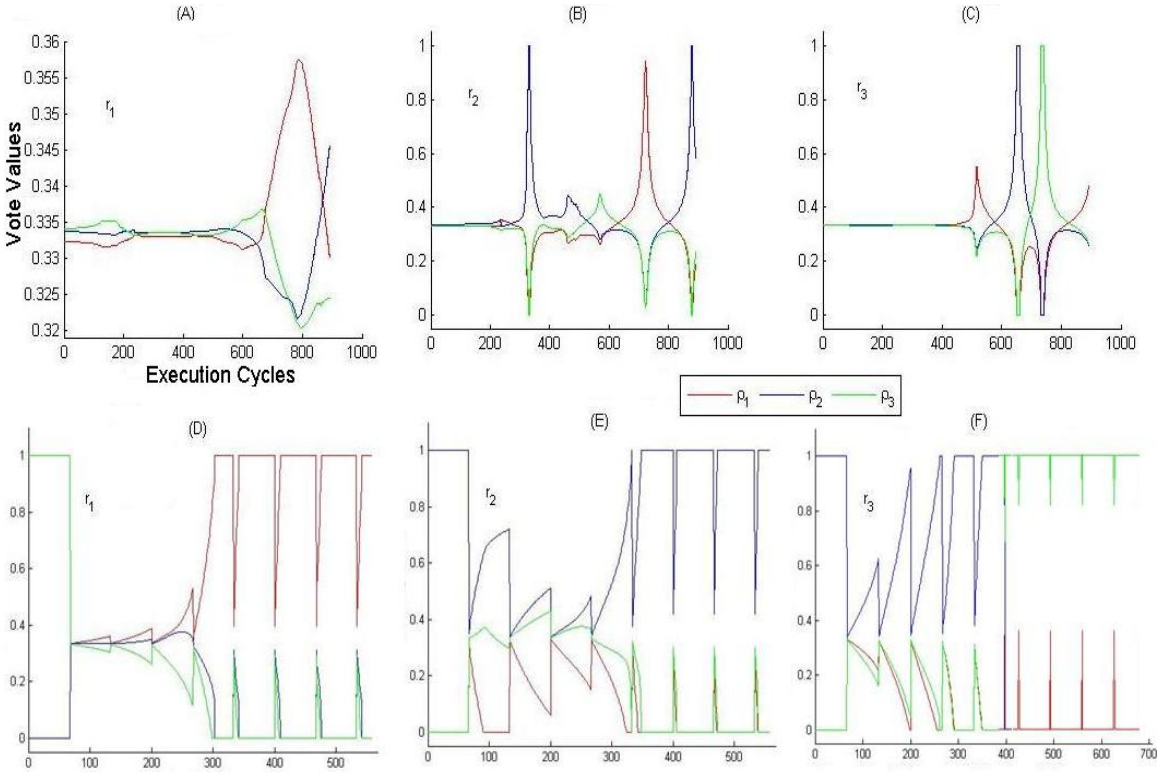

Figure 7.5: The evolution of the votes of pursuers r_1, r_2, and r_3 during the capture phase of the pursuit. Subplots (A), (B), and (C) represent the votes of the default ranking module $\pi_i^t(r_i \mapsto \rho_j)$ of the agents. Subplots (D), (E), and (F) are the votes of the pursuers after the incorporation of the opportunistic ranking module $\omega_i(\rho_j)$ of the pursuers into their decision engine (equation 3.9).

Figure 7.6 provides the distributions of the votes of the pursuers during the capture phase. This figure indicates that the votes of r_1 are mostly distributed on [0.32 0.35] interval. The votes of r_2 and r_3 cover a wider range of values and occasionally reach 0.80 and 1.0, respectively. However, the occurrence of these values are negligible.

The performance of the decision engine of the pursuers changes dramatically after the incorporation of the opportunistic ranking module. In subplots (D), (E), and (F) of Figure 7.5, the peaks and valleys of the curves of votes reveal the active engagement of the decision engine with the relocations of virtual goals. The relocations of these virtual goals are highly dependent on the behavior of the intruder. The behavior of the intruder changes

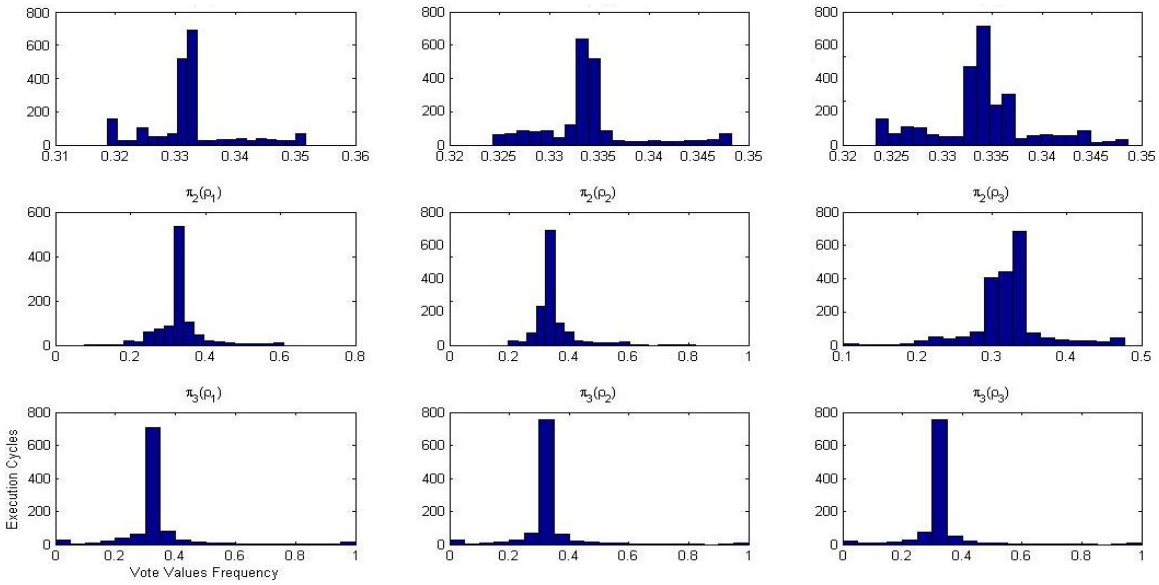

Figure 7.6: Distributions of the votes of default ranking module $\pi_i^t(r_i \mapsto \rho_j)$ of pursuers for a set of virtual goals during the capture phase of a pursuit mission. These histograms provide the frequencies of the votes for the virtual goal $\rho_j \in \mathscr{VG}$. $\pi_i(\rho_j)$ represents the vote of the i^{th} pursuer for the j^{th} virtual goal (equation 3.1).

in response to various events. For example, the intruder changes its navigational direction to avoid collision with a stationary obstacle. A change of the direction in the navigation of the intruder occurs if the intruder detects a pursuer in its vicinity to evade capture. Consequently, the changes of the relocations of the virtual goals result from the variations of their rankings by the decision engines of the pursuers. Although the evolution of the votes of r_1 and r_2 exhibit slow progress between 100 and 300 execution cycles, these votes are consistent with the best choices of these robotic agents. These virtual goals are ρ_1 and ρ_3 for the pursuers r_1 and r_2, respectively. Furthermore, their votes are stabilized after 300 execution cycles. In contrast, it takes longer for r_3 to ascertain its best choice of virtual goal. Subplot (F) in Figure 7.5 shows that the vote of this agent is stabilized after 400 execution cycles. The extended time exhibited by r_3 is due to the choice of the decision engine between 100 and 400 execution cycles. In particular, this agent favors the virtual goal ρ_3 during this time interval. However, r_2 ranks this virtual goal with a higher vote between 100 and 400 execution cycles. This influences the allocation of the virtual goals

during the coordination of the vote profiles of the agents (see section 7.2.3 for details). The distributions of the votes of pursuers after the incorporation of the opportunistic ranking module in the decision process are elaborated in Figure 7.7.

7.2.3 Coordination of the Pursuers

Although Figure 7.2 through Figure 7.7 illustrate the ability of the decision engine of pursuers to elect their best choices, the system requires that every virtual goal is allocated distinctively to a pursuer. For example, Figure 7.2 and Figure 7.4 illustrate a scenario where the decision engine of r_1 and r_2 demand the same virtual goal ρ_3. This results in a situation where the virtual goal ρ_2 is unallocated. This unattended virtual goal provides the intruder with the opportunity to escape if the pursuers are allowed to act solely on the outcome of their respective decision engines. We prevent this situation through the application of the agents votes maximization and the profile matrix permutations strategies (see

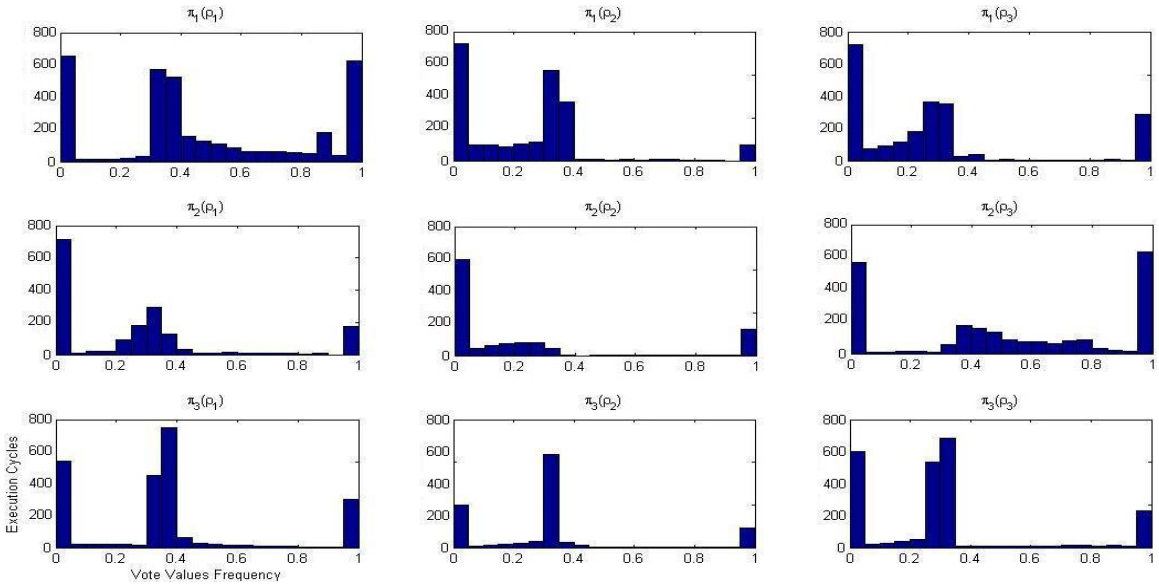

Figure 7.7: Distributions of the votes of the pursuers for a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ after the incorporation of the opportunistic ranking module $\omega_i(\rho_j)$ into decision mechanism during the capture phase of a pursuit mission. These histograms provide the frequencies of the votes for the virtual goal $\rho_j \in \mathscr{VG}$. $\pi_i(\rho_j)$ represents the vote of the i^{th} pursuer for the j^{th} virtual goal (see equation 3.1).

section 4.1, Algorithm 4 and section 4.2, Algorithm 5).

Agents Votes Maximization Strategy

We use this algorithm where the constraint of the virtual goals is $q = 1$ (see Chapter 4, Definition 10 and Lemma 9 for details).[7] This constraint of the virtual goals ensures that every virtual goal is allocated to one pursuer at a given execution cycle. Furthermore, it enables the agents votes maximization strategy to favor the votes of the robotic agents that are highest among all available vote values. Figure 7.8 illustrates the process of the virtual goals allocation to the pursuers using this strategy. The left subplots correspond to the vote

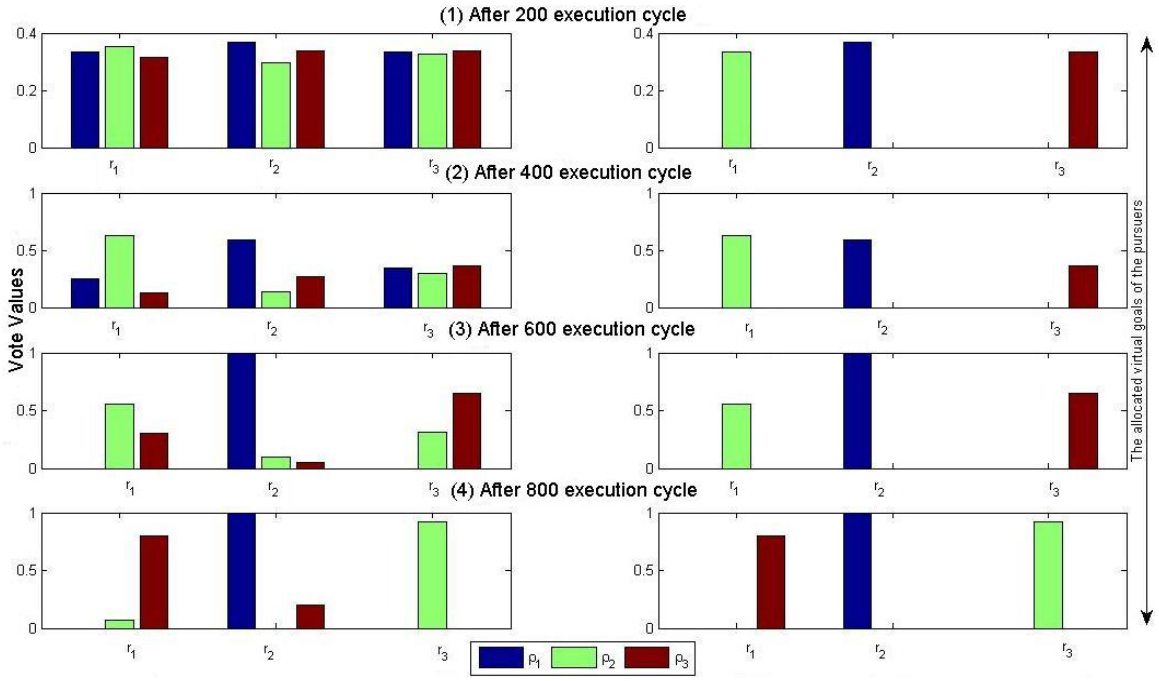

Figure 7.8: The mediation of the vote profiles of the pursuers using the agents votes maximization strategy. The left subplots correspond to the vote profiles of the pursuers r_1, r_2, and r_3 for the set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ in different execution cycles. The right subplots show the allocation of the virtual goals of the pursuer in a given execution cycle. The x-coordinate of these subplots indicate the pursuers $r_i,\ i = 1 \ldots 3$.

[7] We studied the performance of the agents votes maximization strategy under different settings of the constraints of the virtual goals in section 6.4.1.

profiles of the individual pursuers (see Chapter 3, Definition 5). The right subplots show the allocated virtual goals of individual pursuers at different execution cycles. Subplots (1) and (2) are associated with the ambush phase of the pursuit (see section 7.2.1). Subplots (3) and (4) show the result of the agents votes maximization strategies during the capture phase (see section 7.2.2). The x-coordinate of these subplots are labeled r_1, r_2, and r_3 to distinguish the vote profiles and the allocated virtual goal of different agents.

This figure verifies that the allocated virtual goals of the pursuers are distinct. In addition, the virtual goals are allocated based on the highest vote values. For example, the virtual goal ρ_3 (i.e., the blue-colored bar) is allocated to r_2 in subplot (1). This is due to the fact that the vote value of r_2 is higher than the votes of r_1 and r_3 for the virtual goal ρ_2. This characteristic of the agents votes maximization strategy is apparent in every subplot of Figure 7.8. The change of the assignments of r_1 and r_3 are illustrated in subplots (3) and (4). These subplots show that the allocated virtual goals of r_1 and r_3 are exchanged. However, r_2 continues with the same virtual goal ρ_3 throughout the pursuit mission.

Profile Matrix Permutations Strategy

The profile matrix permutations strategy is based on the calculation of the possible permutations of the votes of the pursuers. This strategy utilizes the profile matrix of the pursuers (see Chapter 4, Definition 9) to calculate these permutations. Furthermore, it selects a permutation where the cumulative sum of the votes of agents are maximized (see Chapter 4, Theorem 7). Figure 7.9 illustrates the evolution of the selected permutation in contrast to the possible permutations of the votes of pursuers during the pursuit mission.[8] Although some of the permutations of the votes occasionally rise (e.g., at about 1000 execution cycle), these permutations are dominated by the growth of the cumulative sum of the votes of the selected permutation. Figure 7.9 reveals that the allocated virtual goals of the pursuers determined by the profile matrix permutations strategy are unaltered throughout the mission. This is verified by the linearity of the evolution of the selected permutation in Figure 7.9. Furthermore, the variation of the value of this permutation is negligible in different execution cycles. This results in an allocation strategy where the assignments of the pursuers remain the same throughout the mission. Figure 7.10 illustrates this effect of the linearity

[8]There are six possible permutations of the votes of the pursuers. See section 4.1.1 for an elaborative example.

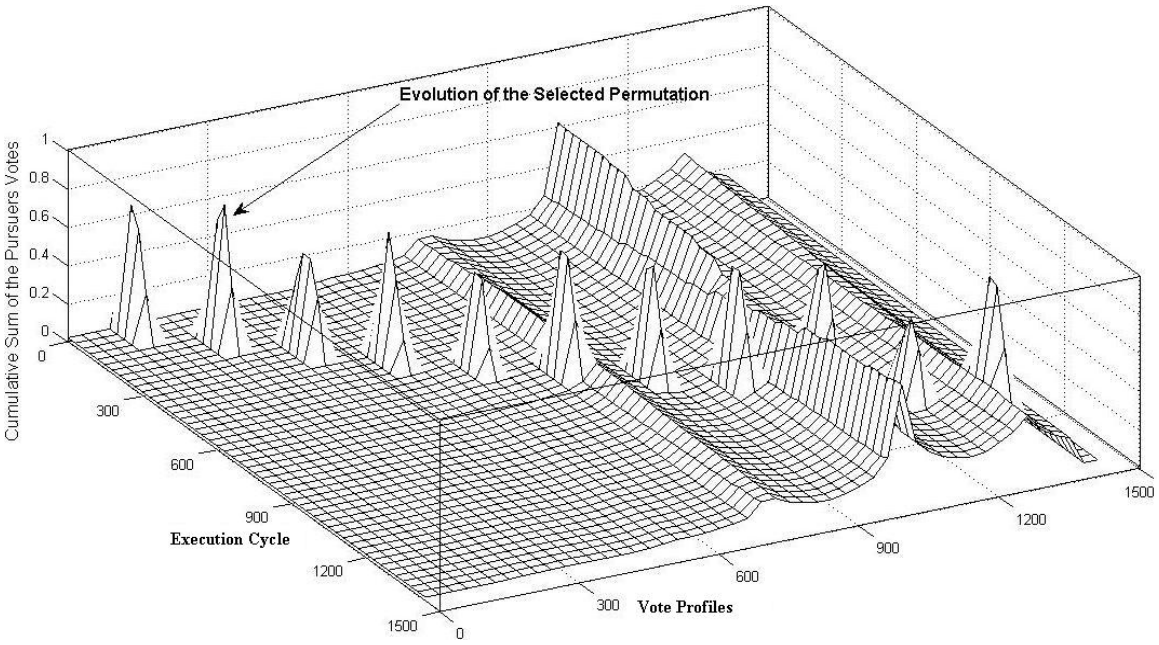

Figure 7.9: The evolution of the selected permutation of the votes of the pursuers.

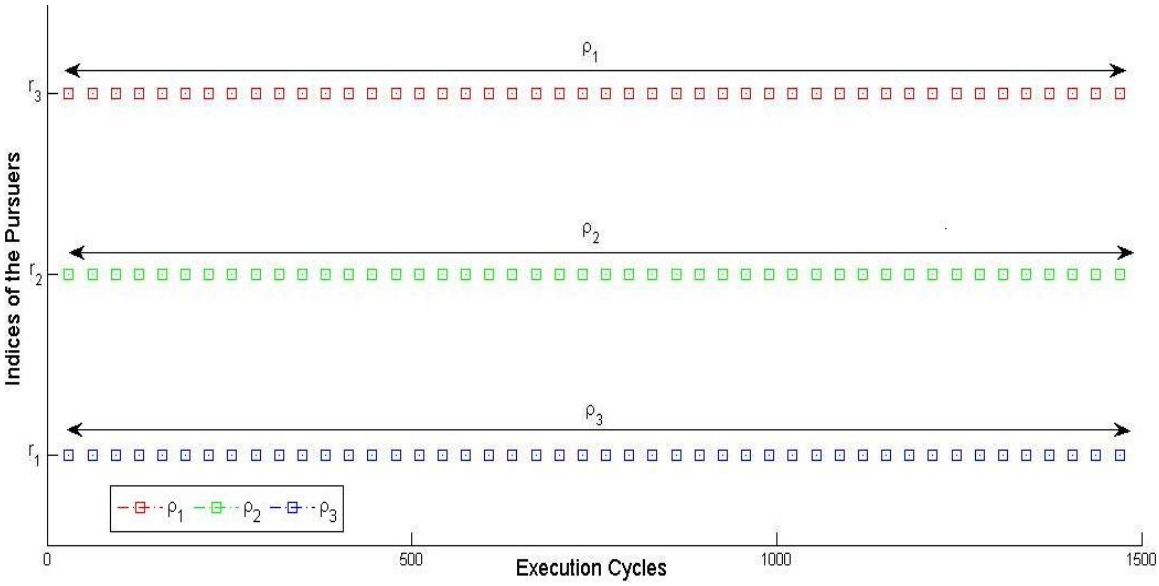

Figure 7.10: The allocated virtual goals of the pursuers based on the profile matrix permutations strategy. These allocated virtual goals are fixed throughout the pursuit mission.

of the evolution of the selected permutation on the assignment of the pursuers. This figure verifies that the allocated virtual goals of the pursuers are fixed throughout the mission. Additionally, Figures 7.9 and 7.10 show that it takes longer in the profile matrix permutations strategy to complete a pursuit mission. A comparison between Figures 7.8 and 7.9 verifies that the agents votes maximization strategy completes a pursuit mission in 800 execution cycles. However, it takes 1500 execution cycles for the profile matrix permutations strategy to capture the intruder.

7.3 Further Analysis

In this section, we study the performance of the agents votes maximization and the profile matrix permutations strategies in contrast to the leader-follower, the prioritization, and the probabilistic approaches.[9] We use the elapsed time, the energy expended, and the distance traveled by the pursuers to compare the performance of these strategies. We use the same simulation setup in all the experiments.

Table 7.1 shows the means and standard deviations for the time expended by the pursuers to complete a pursuit mission using different strategies. In addition, this table provides the percentage of the successful pursuit missions based on these strategies. Table 7.1 indi-

Table 7.1: The mean and the standard deviation of the elapsed time to complete a pursuit mission. The percentage of the successful completion of the pursuit is provided in the last column of the table.

Adapted Strategy	Time (sec.) Mean	STD	Success (%)
Leader-Follower	44.21	4.58	23.8%
Prioritization	38.99	7.96	65.7%
Probabilistic	37.72	6.17	70.3%
Agents Votes Maximization	31.09	7.01	72.9%
Profile Matrix Permutations	41.02	8.67	64.5%

[9]The leader-follower strategy (e.g., Undeger and Polat, 2010) designates one of the robots as the leader of a multi-robot system. These robots follow the strategy that is formulated using the information of the leader (e.g., location information). On the other hand, the prioritization implies deterministic assignments of the tasks at the commencement of a mission. In addition, we use Bayes filter to present the probabilistic approach (see Appendix D).

cates that the agents votes maximization and the probabilistic strategies perform better on average. The mean value entry of this table verifies this improvement on the elapsed time of the pursuers using these two strategies. Furthermore, the standard deviations support the differences in the mean of the elapsed time. Although the profile matrix permutations strategy achieves a better result compared to the leader-follower strategy, its average expended time is more than the agents votes maximizations and the probabilistic strategies. This result is explained by the linearity in the growth of the selected permutation in Figure 7.9. More specifically, this linearity indicates that the choice of the selected permutation by the profile matrix permutations strategy is unaffected by the relocations of the intruder at different execution cycles. Therefore, the allocated virtual goals of the pursuers using the profile matrix permutations resemble the fixed assignments of the prioritization strategy. Table 7.1 indicates that the difference between the mean of elapsed time of the profile matrix permutations and the prioritization strategy is negligible. The mean of the elapsed time of these strategies is within one standard deviation. Hence, the difference in the mean of their elapsed times is not statistically significant. Furthermore, the percentage of successful pursuits in this table supports the similarity of the performance of the profile matrix permutations and the prioritization strategies. However, the prioritization strategy achieves a slightly higher rate of success compared to the profile matrix permutations strategy.

Table 7.2 presents the group-level travel distance of the pursuers in the absence and the presence of obstacles in the environment. This table verifies that the same differences on the performance of these strategies in Table 7.1 continue on the distance traveled by the pursuers. However, the performance of the probabilistic and the agents votes maximization is almost indistinguishable. The mean of the distances traveled by the pursuers using these strategies is within one standard deviation. This result is achieved regardless of the absence or the presence of obstacles in the environment. Therefore, any conclusion on the comparative analysis of the improvement of the distances traveled by the pursuers using these strategies is not warranted. Although the profile matrix permutations exhibits a fixed allocation of the virtual goals during the pursuit (see Figure 7.9), these assignments result in a shorter travel distance compared to the prioritization strategy. This is verified by a comparison of the standard deviation of these strategies.

We mentioned earlier that the pursuers and the intruder are capable of accelerating and decelerating during the pursuit mission. Furthermore, we introduced a maximum limit of $10\frac{m}{s}$ in the change of the velocity of the agents. Therefore, the results presented in Tables 7.1

Table 7.2: The mean, the median, and the standard deviation of the travel distance of the pursuers at the group-level in the absence and the presence of obstacles in the environment.

Adapted	Obstacle-Free			Obstacles		
Strategy	Mean	Median	STD	Mean	Median	STD
Leader-Follower	185.22	163.53	12.34	232.75	196.61	13.49
Prioritization	139.56	119.05	11.16	192.00	164.18	15.25
Probabilistic	116.76	94.13	8.23	161.76	139.13	8.93
Agents Votes Maximization	114.65	94.07	8.34	162.75	141.44	9.40
Profile Matrix Permutations	140.82	123.29	7.79	187.62	167.15	8.56

and 7.2 require further investigation to determine the effect of the different strategies on the energy consumption of the pursuers. These tables show that the agents votes maximization and the probabilistic strategies show a similar trend of the improvement of elapsed time and travel distance. However, there is a significant difference in the energy consumption. Table 7.3 indicates that the result of the allocation of the virtual goals using agents votes maximization strategy yields better average energy consumption. Moreover, the leader-follower strategy does not exhibit a convincing performance. This is due to the fact that the pursuers follow a relatively identical strategy during the pursuit mission. Specifically, pursuers follow the strategy that is inferred based on the information of one of the agents (i.e., the leader of the team). The results of the profile matrix permutations and the prioritization strategies on the energy consumption of the pursuers are relatively similar. However, the prioritization strategy imposes less energy consumption on the pursuers.

Table 7.3: The mean, the median, and the standard deviation of the energy expended by the pursuers at the group-level using different strategies (out of 5000 energy units).

Adapted Strategy	Mean	Median	STD
Leader-Follower	2567.50	1515.40	96.41
Prioritization	2381.80	2568.0	323.19
Probabilistic	1525.50	1557.10	68.73
Agents Votes Maximization	1482.40	1434.9	123.04
Profile Matrix Permutations	2451.72	2484.56	90.89

Figure 7.11 shows the effect of an increase in the velocity of the intruder on the successful completion of the pursuit mission. This figure shows the instances where the velocity of the intruder is 0.50, 0.20, 0.10, 1.0, and 1.1 times the velocity of the pursuers. The leader-follower strategy exhibits an abrupt drop after the velocity of the intruder is set to ≤ 0.50 times of the velocity of the pursuers and continues to fall thereafter. In contrast, the prioritization, the probabilistic, the agents votes maximization, and the profile matrix permutations exhibit a relatively similar trend in the percentage of the successful pursuits.

7.4 Discussion

Research in multi-robot pursuit-evasion demonstrates that three pursuers are sufficient to capture an intruder in a polygonal environment. However, this result requires the confined

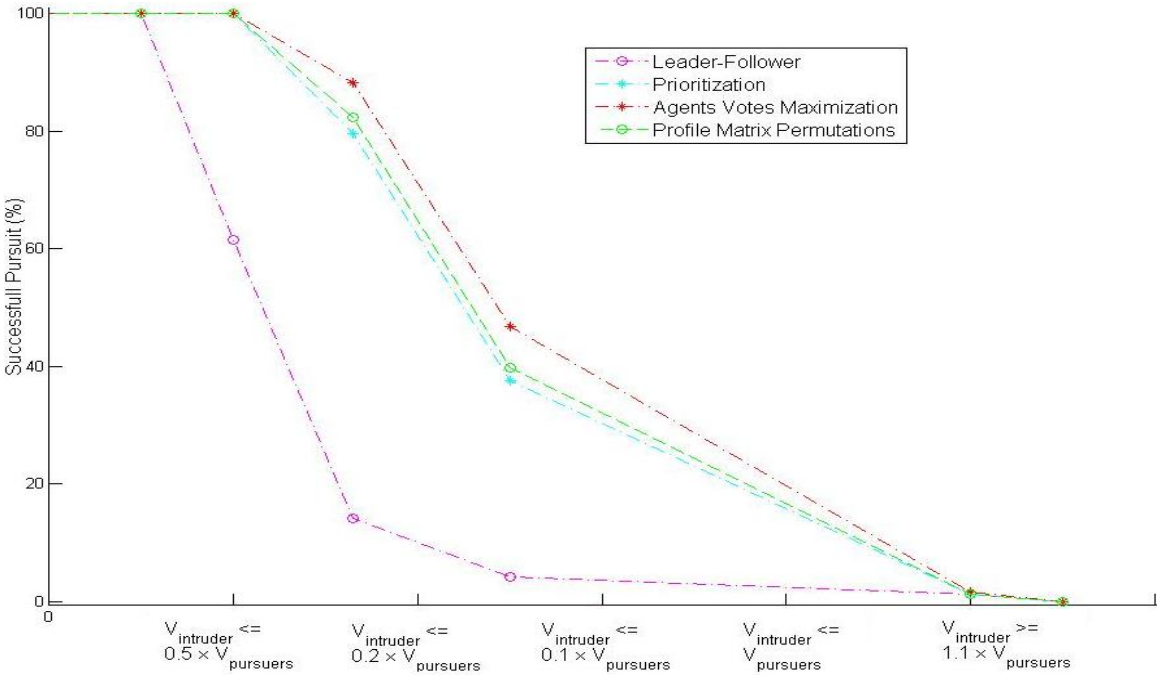

Figure 7.11: The percentage of successful pursuit in conjunction with the increase of the velocity of intruder. $V_{intruder}$ and $V_{pursuers}$ are the velocities of intruder and pursuers, respectively. The x-axis entries indicate the instances where the velocity of intruder is 0.50, 0.20, 0.10, 1.0, and 1.1 times the velocity of pursuers.

of the initial location of the intruder within the convex hull of the locations of the pursuers. In this chapter, we extended this result to demonstrate that this convexity is alleviated through the application of a set of virtual goals that are independent of the locations of the pursuers. These virtual goals are solely calculated using the location information of the intruder such that whose locations confine the intruder within their convex hull at every execution cycle.

We studied the profile matrix permutations and the agents votes maximization strategies to coordinate the independent decisions of the pursuers. This study shows the better performance of the agents votes maximization to the profile matrix permutations. This suggests that the satisfaction of the gain of the pursuers at the group-level is an insufficient assumption to yield an optimal strategy if the group dynamics have a significant effect on the performance of the individuals. More specifically, it compromises the opportunities of some agents in order to maintain the higher gain of the entire group. In contrast, the agents votes maximization strategy results in the performance of the pursuers that is equivalent to the probabilistic framework. This shows the efficiency of the opportunistic ranking module of the external state component of the decision engine mechanism. This module provides a multi-robot system with the capability of inferring decisions where *a priori* information on the state of the mission is absent.

There are various possibilities to extend the result of this study. The isogonic decomposition can be utilized to address the formation among the robotic agents (see Keshmiri and Payandeh, 2011a). We demonstrate the extension of the isogonic decomposition to generate an arbitrary number of the virtual goals (see Appendix B, Theorem 8, and Proposition 1). Networked robotics (e.g., Yao and Gupta, 2009), and military logistics and transportation (e.g., Balch and Arkin, 1998) are among applications where this decomposition approach can be employed.

The behavior of the intruder is an important factor that significantly influences the performance of the pursuers. Although the intruder in this case study performs evasion, its evasive behavior is based solely on the vicinity of the pursuers. Moreover, the intruder evades the pursuers reactively. It is possible to enable the intruder to strategically respond to the detection of the pursuers in the vicinity. Furthermore, the decline of the result of the successful pursuit by using the profile matrix permutations and the agents votes maximization strategies can be analyzed to determine the theoretical bound of the decline in conjunction with the velocity of the intruder.

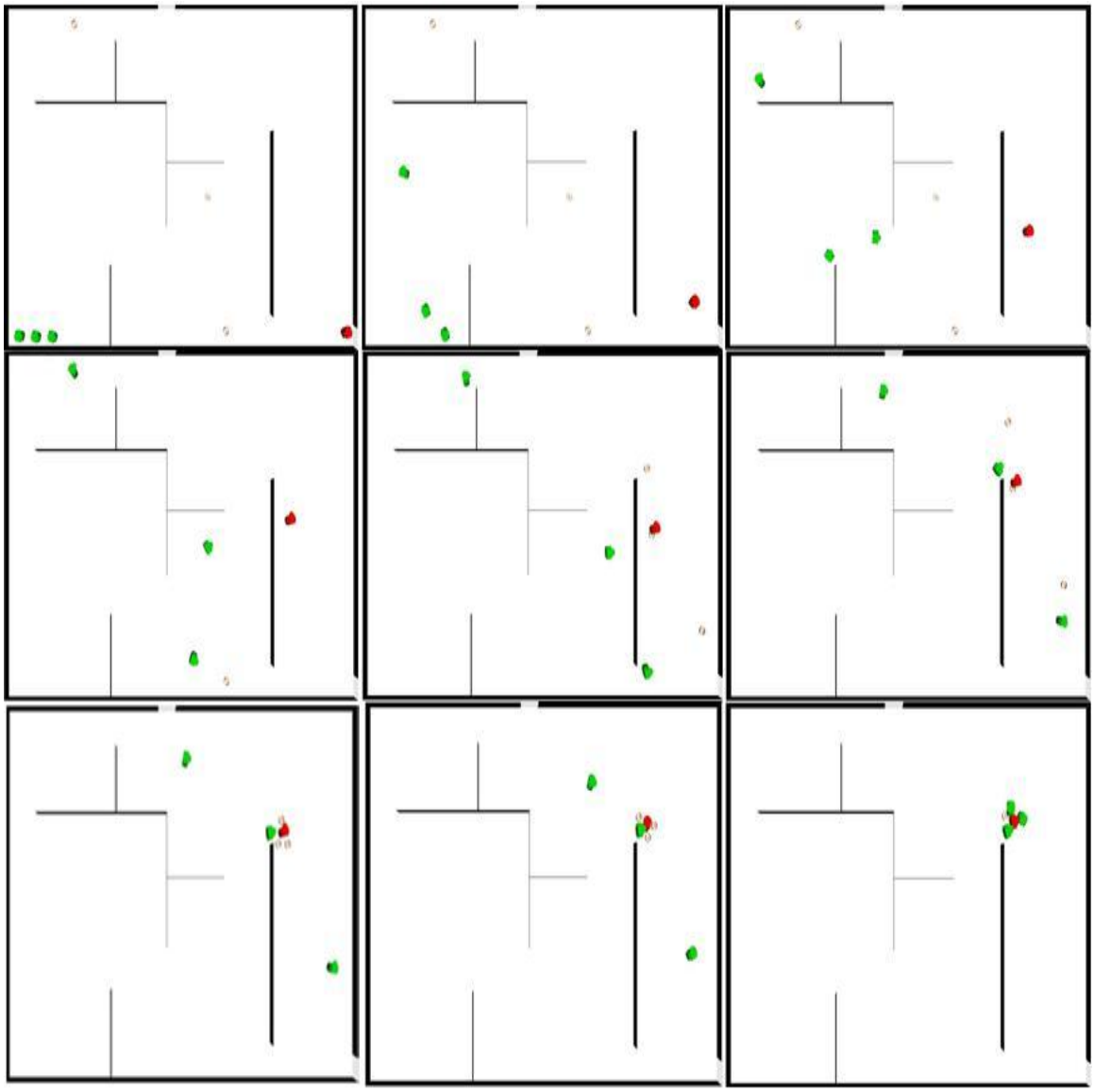

Figure 7.12: Multi-robot, single intruder pursuit.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this thesis we claimed that the cooperation in a complex multi-robot system is achieved implicitly through the mediation of independent decisions. We demonstrated that the proposed approach enables the autonomous agents to proactively contribute to planning, to incremental refinement, as well as to adaptation at the group-level.

We presented a systematic approach to the decomposition process. We introduced the subgrouping to reduce the cardinality of the task space to a number of virtual goals that equals the total number of robotic agents. This has a substantial influence on the decision-making and coordination processes. We demonstrated the ORD and the LSRD techniques utilize the distribution of the robotic agents to transform the decomposition process to an optimization problem. In addition, we presented the isogonic decomposition to reduce the amount of information necessary to decompose the task space using a top-down approach. An important aspect of the decomposition process is its structural flexibility where certain refinement stages are bypassed based on the specification of a mission.

We formulated a decision mechanism to treat the internal and the external states of the robotic agents separately. We demonstrated the capability of the mechanism to calculate the best choice of the virtual goal of the agents at the individual-level. We showed that the complexity of this computation is linear to the cardinality of a set of virtual goals. In addition, we introduced an opportunistic ranking module to the external state component of the decision engine to enable agents to track the evolution of their confidence on available virtual goals. This module empowers the agents to determine their best choices of virtual

goals without presumption of *a priori* information.

We addressed the coordination of these independent decisions through the application of the agents votes maximization and the profile matrix permutations strategies. Using the agents votes maximization strategy, we presented the possibility of the emergence of the cooperation through the dynamical prioritization of the agents with higher votes. Furthermore, we analyzed the effect of the constraints of virtual goals on the inference of the coordination of this strategy. We also demonstrated that the profile matrix permutations provides a multi-robot system with the capability of achieving an optimum allocation strategy based solely on the independent decisions of the autonomous agents. Search and rescue, multi-facility logistic, and security and defense related operations are the areas where the results of this research are applicable. We exemplified these domains of the application of a multi-robot system through the multi-task allocation, the multi-location rendezvous, and the pursuit-evasion scenarios.

We showed that the combination of subgrouping and profile matrix permutations enables a multi-robot system to perform a multi-task allocation in a scale that is significantly larger than one-to-one mapping of the robots and the subtasks. Furthermore, we demonstrated that the approach achieves an optimum allocation strategy to resolve the exponential growth of the conventional approaches.

We studied the performance of the linear decomposition and the agents votes maximization strategy in a multi-robot, multi-location rendezvous scenario. We demonstrated that the utilization of distributional information from robotic agents enables the system to compute rendezvous locations to minimize the travel distance at the group-level. Furthermore, we showed that the performance of the approach is unaffected by the constraints imposed on the number of attendees of the rendezvous locations.

We extended the result in pursuit-evasion research to alleviate the necessary condition of the confinement of the initial location of the intruder within the convex hull of the locations of pursuers. We demonstrated that the isogonic decomposition achieves this confinement through the computation of a set of virtual goals that are independent of the locations of the pursuers. We also showed that the location information of the intruder is sufficient to compute the set of virtual goals. Additionally, the study reveals that the agents votes maximization achieves a better results in coordinating the pursuers compared to the profile matrix permutations. This observation challenges the concept of optimization in the allocation of agents at the group-level if the dynamic of the delegated mission has a significant

effect on the performance of the individuals.. More specifically, the profile matrix permutations strategy compromises the opportunities of some of the pursuers to maintain the higher gain of the entire group. This makes the system inflexible to respond to the change of the behavior of the intruder. In contrast, the results of the agents votes maximization strategy is equivalent to the probabilistic framework. This suggests the efficiency of the opportunistic ranking module in the external state component of the decision engine to infer decisions without dependency on the priors to calculate the votes.

8.2 Future Work

In this research the continuity and accuracy of the location information of tasks to calculate the set of virtual goals is the basic assumption in the decomposition process. However, this information is error prone in real-life problem domains. Moreover, it is possible for a system to acquire this information in discontinuous intervals. An extension of this research is to study the effect of such uncertainties in the location information of tasks on the decomposition process. For example, it is possible to incorporate a tracking mechanism[1] to estimate these locations based on the available imperfect information.

The formulation of decision engine in the current research assumes that the set of virtual goals is received synchronously and without any further delays. Moreover, the set is assumed to form the common knowledge of the robotic agents during the voting process. The instantaneous availability of a set of virtual goals can be challenged through the introduction of communication failure between the decomposition mechanism and the decision engines in different decision cycles.

The present implementation of the decision engine is not constrained by parameters such as level of acuteness or the preemptive preferences among the subtasks. This simplification of decision-making and subsequently the coordination of a multi-robot system is not reflected in scenarios where the ordering and speed of transition among subtasks significantly affects the results of the operation. It is possible to extend the external state component of the decision engine to minimize the total travel time or the cost of the trade-off of energy and time in addition to the travel distance. This modification of the decision engine requires the inclusion of several other factors to calculate the votes. These factors vary from the velocity

[1]Kalman and particle filters are the examples of such tracking mechanisms (see Maybeck, 1990; Metropolis and Ulam, 1949).

bound imposed on an agent, to the time constraints and the ordering of the subtasks.

The opportunistic ranking module of the external state component can be extended to examine the human-robots collaboration and interaction. For example, the opportunistic ranking module can provide the robotic agents with additional information from the human agent to influence the computation of their vote profiles. This information can guide or redirect the attention of the robots to the specific aspects of a mission that is more important to the third-party human observer. This research can be extended to analyze the behavior of decision engine and coordination strategies under the starvation condition where the available energy of some of the agents is insufficient to reach all the subtasks.

The computation of all possible permutations of votes results in the exponential growth of complexity in the coordination strategy inferred by the profile matrix permutations. This growth of complexity can be improved through the allocation of subgroups to the clusters of robots where the number of clusters ≤ 5. The incorporation of a learning process that enables the profile matrix permutations to eliminate the cumulative sum of the votes that are persistently below a threshold is another alternative to improve the complexity of this strategy. A challenging problem that can be studied in the future is the introduction of a number of mediators that perform the allocation in parallel and in a distributed fashion.

It is possible to enhance the cost of the relocation of robotic agents to address the uncertainties that exist in their location information. Alternatively, these costs can be formulated to estimate the contingent correlation between the cost of the relocation and their dynamics. These costs can be utilized to incorporate the effect of environmental conditions, such as the type of the terrain, on the relocations of the agents.

The pursuit scenario can be enhanced to study the performance of the system in response to a more sophisticated intruder. The behavior of the intruder can be improved to exhibit complicated evasive strategies. The rendezvous problem can be studied in presence of a number of service robots. This modification adds an additional level of complexity to the coordination of the allocation of the rendezvous regions to these agents.

Bibliography

Aigner, M., and Fromme, M. (1984). A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1–12.

Alspach, B. (2004). Searching and sweeping graphs: a brief survey. *Matematiche*, 59(1):5–37.

Amari, S., and Kawanabe, M. (2002). Basic concepts and analysis in EIV modeling. tls and its improvements by semiparametric approach. In Huffel, S. V. and Lemmerling, P., editors, *Total least squares and errors-in-variables modeling : analysis, algorithms and applications*. Kluwer Academic Publishers, Dordrecht, Netherland.

Amigoni, F., Basilico, N., Gatti, N., Saporiti, A., and Troiani, S. (2010). Moving game theoretical patrolling strategies from theory to practice: an usarsim simulation. In *IEEE International Conference on Robotics and Automation (ICRA10), Milan, Italy*, pages 426–431.

Anderson, C., and Franks, N. R. (2001). Teams in animal societies. *Behavioral Ecology*, 12(5):534–540.

Andersson, M., and Sandholm, T. (2000). Contract type sequencing for re-allocative negotiation. In *International Conference on Distributed Computing Systems*.

Assaf, D., and Zamir, S. (1985). Optimal sequential search: a Bayesian approach. *Annals of Statistics*, 13(3):1213–1221.

Arkin, R. (1998). *Behavior-based Robotics*. The MIT Press, Cambridge.

Atay, N., and Bayazit, B. (2006). Mixed-integer linear programming solution to multi-robot task allocation problem. *Technical Report WUCSE-2006-54, Department of Computer Science and Engineering, Washington University*.

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

Bargiela, A., and Hartley, J. K. (1993). Orthogonal linear regression algorithm based on augmented matrix formulation. *Journal of Computers and Optimizations Research*, 20(9):829–836.

Basar, T., and Olsder, G. J. (1999). *Dynamic noncooperative game theory*. Philadelphia: Society for Industrial Mathematics.

Beil, A. C., and Vaughan, R. (2009). Adaptive mobile charging stations for multi-robot systems. In *International Conference on Intelligent Robots and Systems, (IROS), St. Louis, MO, USA*, pages 1363–1368 .

Berhault, M., Huang, H., Keskinocak, P., Koenig, S., Elmaghraby, W., Griffin, P., and Kleywegt, A. (2003). Robot exploration with combinatorial auctions. In *IEEE/RSJ International Conference Intelligent Robots and Systems (IROS), Atlanta, GA, USA*, pages 1957–1962.

Bertsekas, D. P. (1992). The auction algorithm for assignment and other network flow problems: a tutorial introduction. *Computational Optimization and Applications*, 1(1):7–66.

Bevington, P. R., and Robinson, D. K. (2003). *Data reduction and error analysis*. McGraw-Hill, 5th edition, Boston.

Boltyanski, V., Martini, H., and Soltan, V. (1999). *Geometric methods and optimization problems*. Kluwer Academic Publishers, Boston.

Bondy, J. A., and Murty, U. S. R. (1976). *Graph theory with applications*. New York : Americn Elsevier.

Boots, B., Okabe, A., and Sugihara, K. (1994). Nearest neighborhood operations with generalized voronoi diagrams: a review. *International Journal of Geographical Information Systems*, 8(1):43–71.

Bopardikar, S. D., Bullo, F., and Hespanha, J. P. (2007). Sensing limitations in the lion and man problem. In *IEEE American Control Conference, Santa Barbara, USA*, pages 5958–5963 .

Borenstein, J., Everett, B., and Feng, L. (1996). *Navigating Mobile Robots: Systems and Techniques*. Wellesley, MA.

Botelho, S., and Alami, R. (2009). M^+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *IEEE International Conference on Robotics and Automation, (ICRA), Toulouse, France*, pages 1234–1239.

Brown, K. L., and Shoham, Y. (2008). *Essentials of game theory*. Morgan & Claypool Publisgers.

Bourgeois, F., and Lassalle, J. C.(1971). An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12):802–804.

Butterfield, J., Jenkins, O. C., and Gerkey, B. (2008). Multi-robot markov random fields. In 7^{th} *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1211–1214 .

Calamai, P. H., and Conn, A. R. (1980). A stable algorithm for solving the multifacility location problem involving euclidean distances. *SIAM Journal on Scientific and Statistical Computing*, 1:512–526.

Campbell, A., and Wu, A. S. (2011). Multi-agent role allocation: issues, approaches, and multiple perspectives. *Autonomous Agents and Multi-Agent Systems*,23:317–355.

Cao, Y. U., Fukunaga, A. S., and Kahng, A. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27.

Chalkiadakis, G. (2003). Multiagent reinforcement learning: Stochastic games with multiple learning players. *Department of Computer Science, Univeristy of Toronto, Technical report, March 2003. [Online]. Available: www.cs.toronto.edu/gehalk/DepthReport/DepthReport.ps* .

Chalkiadakis, G., and Boutilier, C. (2003). Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Second International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS)*.

Charniak, E. (1991). Bayesian network without tears. *AI Magazine*, 12(4):50–63.

Chatterjee, S., and Price, B. (1991). *Regression Analysis by Example*. New York: Wiley series in probability and mathematical statistics, 2^{nd} edition.

Chung, C. F., and Furukawa, T. (2009). Coordinated pursuer control using particle filters for autonomous search-and-capture. *Robotics and Autonomous Systems*, 57(6–7):700–711.

Chung, T. H., and Hollinger, G. A. (2011). Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31(4):299–316.

Cortes, J., Martinez, S., and Bullo, F. (2006). Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transaction on Automatic Control*, 51(8):1289–1298 .

Dahl, T. S., Mataric, M. J., and Sukhatme, G. S. (2009). Multi-robot task allocation through vacancy chain scheduling. *Robotics and Autonomous Systems*, 57(6–7):647–687.

Devore, J. L. (2004). *Probability and statistics for engineering and the sciences*. Thomson (Brooks/Cole), 6^{th} edition. Pacific Grove, California.

Dias, M. B., and Stentz, A. (2001). A market-based approach to multi-robot coordination. *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-01-26* .

Dias, M. B., and Stentz, A. (2002). Opportunistic optimization for market-based mulitrobot control. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Pittsburgh, PA, USA*, pages 2714–2720.

Dias, M. B., Goldberg, D., and Stentz, A. (2003). Market-based multirobot coordination for complex space applications. In 7^{th} *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*.

DiasAdversarial2005 Dias, M. B., Browning, B., Veloso, M. M., and Stentz, A. (2005). Dynamic heterogeneous robot teams engaged in adversarial tasks. *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-14* .

Dias, M. B., Zlot, R., Kalra, N., and Stentz, A. (2006). Market-based multirobot coordination: a survey and analysis. *Proceedings of the IEEE*, 94:(7):1257–1270 .

Dudek, G., and Roy, N. (1997). Multi-robot rendezvous in unknown environments or, what to do when you are lost at the zoo. In *AAAI National Conference Workshop on Online Search, Providence, Rhode Island*.

Dudek, G., Jenkin, M., and Milios, E. (2002). A taxonomy of multirobot systems. In Balch, T. and Parker, L., editors, *Robot Teams: from Diversity to Polymorphism*, A. K. Peters, Natick, MA, pages 3–22.

Fomin, F. V., and Thilikos, D. M. (2008). An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3):236–245.

Furukawa, T., Bourgault, F., Lavis, B., and Durrant-Whyte, H. F. (2006). Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets. In *IEEE International Conference on Robotics and Automation (ICRA), Sydney, NSW*, pages 2521–2526 .

Gerkey, B. P., and Mataric, M. J. (2002). Sold!: Auction methods for multi-robot control. *IEEE Transaction on Robotics and Automation (Special Issue on Multi-Robot Systems)*, 18(5):758–768.

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

Gerkey, B. P., and Mataric, M. J. (2004b). On role allocation in robocup. *Lecture Notes in Computer Science, RoboCup 2003: Robot Soccer World Cup VII*, pages 43–53 .

Gravetter, J. F., and Wallnau, L. B. (2008). *statistics for the behavioral sciences*. Belmont, CA, USA: Wadsworth, Cengage Learning.

Guo, J., Yan, G., and Lin, Z. (2010). Cooperative control synthesis for moving-target-enclosing with changing topologies. In *IEEE International Conference on Robotics and Automation (ICRA), Hangzhou, China*, pages 1468–1473 .

Harmati, I., and Skrzypczyk, K. (2009). Robot team coordination for target tracking using fuzzy logic controller in game theoretic framework. *Robotics and Autonomous Systems*, 57(1):75–86.

Isler, V., Kannan, S., and Khanna, S. (2005). Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 21(5):875–884.

Isler, V., and Karnad, N. (2008). The role of information in the cop-robber game. *Theoretical Computer Science, Special issue on graph searching*, 399(6):179–190.

Jankovic, V. (1978). About a man and lions. *Matematicki Vesnik*, 2:359–361.

Jeffrey, A. (2005). *Mathematics for engineers and scientists*. 6^{th} edition Chapman & Hall/CRC Press, Boca Raton.

Kamal, S., Gani, M., and Seneviratne, L. (2010). A game-theoretic approach to non-cooperative target assignment. *Robotics and Autonomous Systems*, 58(8):955–962.

Karaboga, D., and Akay, B. (2009). A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, 31(1–4):61–85.

Karavelas, M. I. (2004). A robust and efficient implementation for the segment Voronoi diagram. In 1^{st} *International Symposium on Voronoi Diagrams in Science and Engineering*, pages 51–62.

Keshmiri, S., and Payandeh, S. (2011). Multi-robot target pursuit: towards an opportunistic control architecture. In *Eighth International Multi-Conference on Systems, Signals & Devices (SSD), Sousse, Tunisia*, pages 22–25.

Keshmiri, S., and Payandeh, S. (2011a). A Centralized Framework to Multi-Robots Formation Control: Theory and Application. *Lecture Notes in Computer Science*, pages 85–98.

Kim, D. H., and Kim, J. H. (2002). A real-time limit-cycle navigation for fast mobile robots and its application to robot soccer. *Autonomous and Robotics Systems*, 42(1):17–30.

Kim, T. H., and Sugie, T. (2007). Cooperative control for target-capturing task based on a cyclic pursuit strategy. *Automatica*, 43(8):1426–1431.

Koopman, B. O. (1956a). The theory of search. Part I. Kinematic bases. *Operations Research*, 4(5):324–346.

Koopman, B. O. (1956b). The theory of search. Part II. Target detection. *Operations Research*, 4(5):503–531.

Kopparty, S., and Ravishankar, C. V. (2005). A framework for pursuit evasion games in R_n. *Information Processing Letters*, 96(3):114–122.

Kose, H., Tatlidede, U., Mericli, C., Kaplan, K., and Akin, H. L. (2004). Q-learning based market-driven multi-agent collaboration in robot soccer. In *Turkish Symposium Artificial Intelligence and Neural Networks*.

Krieger, M. J. B., and Billeter, J. B. (2000). The call of duty: Self-organized task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30:65–84.

Kube, C. R., and Zhang, H. (1996). The Use of perceptual cues in multi-robot box-pushing. In *IEEE International Conference on Robotics and Automation (ICRA), Edmonton, AB*, pages 2085–2090.

Kuffner, J. J., and LaValle, S. M. (2000). RRT-connect: and efficient approach to a single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA), Stanford Univ., CA, USA*, pages 995–1001.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2(1–2):83–97.

Kupitz, Y., and Martini, H. (1997). Geometric aspects of the generalized Fermat-Torricelli problem. *Bolyai Society mathematical studies*, 6:55–129.

Kutner, M. H., Nachtsheim, C. J., Neter, J., and Li, W. (2005). *Applied Linear Statistical Models*. Boston: McGrow-Hill Irwin, 5^{th} edition.

Lagoudakis, M., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, S., Koenig, S., Meyerson, C. A., and Jain, S. (2005). Auction-based multi-robot routing. In *Robotics: Science and System*.

Lemaire, T., Alami, R., and Lacroix, S. (2004). A distributed tasks allocation scheme in multi-UAV context. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Toulouse, France*, pages 3622–3627.

Lein, A., and Vaughan, R. (2008). Adaptive multi-robot bucket brigade foraging. In Bullock, S., Noble, J., Watson, R., and Bedau, M. A. editors, *Artificial life XI: proceedings of the* 11^{th} *International Conference on the Simulation and Synthesis of Living Systems*. Cambridge: MIT Press, pages 337–342.

Levy, R., and Rosenschein, J. (1992). A game theoretic approach to the pursuit problem. In 11^{th} *International Workshop on Distributed Artificial Intelligence*, pages 195–213.

Lin, A., and Anderson, B. D. (2003). The multi-agent rendezvous problem. In 42^{nd} *IEEE Conference on Decision and Control, New Haven, CT, USA*, pages 1508–1513.

Litus, Y., Zebrowski, P., and Vaughan, R. (2009). A distributed heuristic for energy-efficient mulitrobot multiplace rendezvous. *IEEE Transactions on Robotics*, 25(1):130–135.

Litus, Y., Vaughan, R., and Zebrowski, P. (2007). The frugal feeding problem: energy-efficient: Multi-robot, multi- place rendezvous. In *IEEE International Conference on Robotics and Automation, (ICRA), Roma, Italy*, pages 27–32.

Liu, L., and Shell, D. A. (2011). Assessing optimal assignment under uncertainty: an interval-based algorithm. *Robotics Research*, 30(7):936–953 .

Martinoli, A. (1999). *Swarm intelligence in autonomous collective robotics: from tools to the analysis and synthesis of distributed control strategies*. PhD thesis, EPFL — cole Polytechnique Fdrale de Lausanne (EPFL). Lausanne, EPFL, 1999.

Maybeck, P. S. (1990). *The Kalman filter: an introduction to concepts*. Autonomous Robot Vehicles, Springer, Verlag

Megiddo, N., and Tamir, A. (1983). Finding least-distances line. *Siam Journal of Algebraic and Discrete Methods*, 4(2):207–211.

Mei, Y., Lu, Y. H., Hu., Y. C., and Lee, C. (2005). A case study of mobile robot's energy consumption and conservation techniques. In 12^{th} *International Conference on Advanced Robotics (ICAR), Indianapolis, IN, USA*, pages 492–497 .

Melloy, B. J., and Cavalier, T. M. (1991). An iterative linear programming solution to the euclidean regression model. *Journal of Computers and Optimizations Research*, 18(8):655–661.

Metropolis, N., and Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341.

Morters, P., and Peres, Y. (2010). *Brownian Motion*. Cambridge University Press, United Kingdom.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Society for Industrial and Applied Mathematics*, 5(1):32–38.

Munoz, A., Sempe, F., and Drogoul, A. (2002). Sharing a charging station in collective robotics. *8, rue du Capitaine Scott, 75015 Paris, Tech. Rep. LIP6 2002/026* .

Nair, R., Ito, T., Tambe, M., and Marsella, S. (2002). Task allocation in the rescue simulation domain: a short note. In *Lecture Notes in Computer Science, RoboCup-2001: Fifth Robot World Cup Games and Conference*, pages 1–22.

Nanjanath, M., and Gini, M. (2010). Repeated auctions for robust task execution by a robot team. *Robotics and Autonomous Systems*, 58(7):900–909.

Neter, J., Wasserman, W., and Kutner, M. H. (1990). *Applied statistical models: regression, analysis of variance, and experimental designs*. Homewood, 3^{rd} edition, Irwin.

Ngo, T. D., Raposo, H., and Schioler, H. (2008). Multi-agent Robotics: toward energy autonomy. *Artificial Life and Robotics*, 12(1–2):47–52.

Nowakowski R., and Winkler, P. (1983). Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235–239.

Oh, S., and Zelinsky, A. (2000). Autonomous battery recharging for indoor mobile robots. In *Australian Conference on Robotics and Automation, [online]. Available: citeseer.ist.psu.edu/oh00autonomous.html*.

Okabe, A., and Boots, B. (2000). *Spatial Tessellations: concepts and applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics.

Okabe, A., and Suzuki, A. (1997). Locational optimization problems solved through voronoi diagrams. *European Journal of Operational Research*, 98(3):445–456.

Oster, G. F., and Wilson, E. O. (1978). *Caste and Ecology in the social insects*. Princeton, NJ: Princeton University Press.

Ostergaard, E., Sukhatme, G. S., and Mataric, M. J. (2001). Emergent bucket brigading – a simple mechanism for imrpoving performance in multi-robot constarined-space foraging tasks. In *International Conference on Autonomous Agents*, pages 2219–2223.

Parker, C. A. C., and Zhang, H. (2008). Consensus-based task sequencing in decentralized multiple-robot systems using local communication. In *IEEE/RSJ international conference on intelligent robots and systems (IROS), Edmonton, AB, Canada*, pages 1421–1426 .

Parker, C. A. C., and Zhang, H. (2010). Collective unary decision-making by decentralized multiple-robot systems applied to the task-sequencing problem. *Swarm Intelligence*, 4(3):199–220.

Parker, C. A. C., and Zhang, H. (2011). Biologically inspired collective comparisons by robotic swarms. *The International Journal of Robotics Research*, 30(5):524–535.

Parker, L. E. (1997). L-alliance: Task-oriented multi-robot learning in behavior-based systems. *Advanced Robotics, Selected Papers from IROS'96*, 11.

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

Pini, G., Brutschy, A., Birattari, M., and Dorigo, M. (2011a). Task partitioning in swarms of robots: Reducing performance losses due to interference at shared resources. *Lecture Notes in Electrical Engineering*, 85(3):217–228.

Pini, G., Brutschy, A., Frison, M., Roli, A., Dorigo, M., Birattari, M. (2011b). Task partitioning in swarms of robots: an adaptive method for strategy selection. *Swarm Intelligence*, 5(3–4):283–304.

Rabideau, G., Estlin, T., Chien, S., and Barrett, A. (1999). A comparison of coordinated planning methods for cooperating rovers. In *AIAA 1999 Space Technology Conference, New York, NY, USA*.

Rus, D., and Vona, M. (1999). Self-reconfiguration planning with compressible unit modules. In *IEEE International Conference on Robotics and Automation, (ICRA), Detroit, Michigan, USA*, pages 2513–2520.

Salemi, B., and Shen, W. M. (2001). Hormone-controlled metamorphic robots. In *IEEE International Conference on Robotics and Automation, (ICRA), Seoul, Korea*, pages 4194–4199.

Sandholm, T. (2002). Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1–2):1–54.

Schlude, K. (2003). From robotics to facility location: Contraction functions, weber point, convex core. *Technical Report 403, Computer Science, ETHZ* .

Schneider, J., Apfelbaum, D., Bagnell, D., and Simmons, R. (2005). Learning opportunity costs in multi-robot market based planner. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Pittsburgh, PA, USA*, pages 1151–1156 .

Service, T. C., and Adams, J. A. (2011). Coalition formation for task-allocation: theory and algorithms. *Robotics and Autonomous Systems*, 22(2):225–248.

Shell, D. A., and Mataric, M. J. (2006). On foraging strategies for large-scale multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Los Angeles, CA, USA*, pages 2717–2723 .

Shoham, Y., and Brown, K. L. (2009). *Multiagent systems : algorithmic, game-theoretic, and logical foundations*. New York : Cambridge University Press.

Silverman, M., Nies, D. M., Jung, B., and Sukhatme, G. S. (2002). Staying alive: a docking station for autonomous robot recharging. In *International Conference on Robotics and Automation (ICRA), El Segundo, CA, USA*, pages 1050–1055

Stone, P. (2007). *Intelligent autonomous robotics: a robot soccer case study*. Morgan & Claypool Publishers' Series.

Talay, S. S., Balch, T. R., and Erdogan, N. (2011). A generic framework for distributed multirobot cooperation. *Autonomous Agents and Multi-Agent Systems*, 63:323–358.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001). Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1–2):99–141.

Thrun, S., Burgard, W., Fox, D. (2006). *Probabilistic Robotics*. The MIT Press, Cambridge, Massachusetts, USA.

Thunberg, J., and Ogren, P. (2010). An iterative mixed integer linear programming approach to pursuit evasion problems in polygonal environment. In *International Conference on Robotics and Automation (ICRA), Stockholm, Sweden*, pages 5498–5503.

Tou, J. T., and Gonzalez, R. C. (1974). *Pattern recognition principles.* Massachusetts: Addison-Wesley Publishing Company, Inc.

Tovey, C., Lagoudakis, M. G., Jain, S., and Koenig, S. (2005). The generation of bidding rules for auction-based robot coordination. In *Multi-Robot Systems. From Swarms to Intelligent Automata*, 3:3–14.

Undeger, C., and Polat, F. (2010). Multi-agent real-time pursuit. *Autonomous Agents and Multi-Agent Systems*, 21(1):69–107.

Walker, J. H., and Wilson, M. S. (2011). Task allocation for robots using inspiration from hormones. *Adaptive Behavior*, 19(3):208–224 .

Washburn, A. R. (1983). Search for a moving target: the FAB algorithm. *Operations Research*, 31(4):739–751.

Werger, B. B., and Mataric, M. J. (2001). broadcast of local eligibility for multi-target observation. *Distributed Autonomous Robotic Systems 4, L. E. Parker, G. Bekey, and J. Barhen, Editors, Springer-Verlag, New York*, pages 347–356.

Wolfstetter, E. (1996). Auctions: an introduction. *Economic Surveys*, 10(4):367–420.

Yao, Z., Gupta, K. (2009). Backbone-based connectivity control for mobile networks. In *IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan*, pages 1133–1139 .

Zebrowski, P., and Vaughan, R. (2005). Recharging robot teams: a tanker approach. In *International Conference on Advanced Robotics (ICAR)*, pages 803–810 .

Zebrowski, P., Litus, Y., and Vaughan, R. (2007). Energy efficient robot rendezvous. In 4^{th} *Canadian Conference on Computer and Robot Vision, Montreal, QC, Canada*, pages 139–148.

Zhang, Q., Yang, Y., and Li, Y. (2002). A multi-agent cooperative system of soccer robot. In *Proceedings of the* 4^{th} *World Congress on Intelligent Control and Automation, Guangzhou, China*, pages 510–514.

Zhenwang, Y., and Gupta, K. (2009). Back-bone connectivity control for mobile networks. In *IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan*, pages 1133–1139.

Zlot, R., Stentz, A., Dias, M. B., and Thayer, S. (2002). Multi-robot exploration controlled by a market economy. In *IEEE International Conference on Robotics and Automation (ICRA), Pittsburgh, PA, USA*, pages 3016–3023.

Appendix A

Linear Regression

A.1 Introduction

Regression analysis is a method of statistical analysis that investigates the relationship between two or more variables that are related in a deterministic fashion (see Devore, 2004). The primarily of regression analysis and its corresponding techniques is to model and analyze numerical data comprising values of a dependent variable (also called the response variable) and one or more independent variables (also referred to as predictor or explanatory variables).

Simple linear regression is a tool in which the expected value of dependent variable is assumed to be a linear function of explanatory variables. It uses the method of least squares to interpret the relationship between the explanatory and response variables by fitting a straight line that minimizes the sum of the squares of vertical distances of data points from the best fitting line. Deviations are generally measured along the y-coordinate (see Jeffrey, 2005). The best fit in least squares is the instance of the model for which the sum of squared residuals has its least value. A residual is the difference between an observed value and the predefined value of the model. To represent data graphically (e.g., scatter plot) the independent variable is shown along the x-coordinate whereas the dependent variable forms the y-coordinate values.

Figure A.1 illustrates the concept of least squares linear regression. Blue-colored points are the observed data. The red-colored line in bottom subplot is the least square fit, given the distribution of data points in the top subplot.

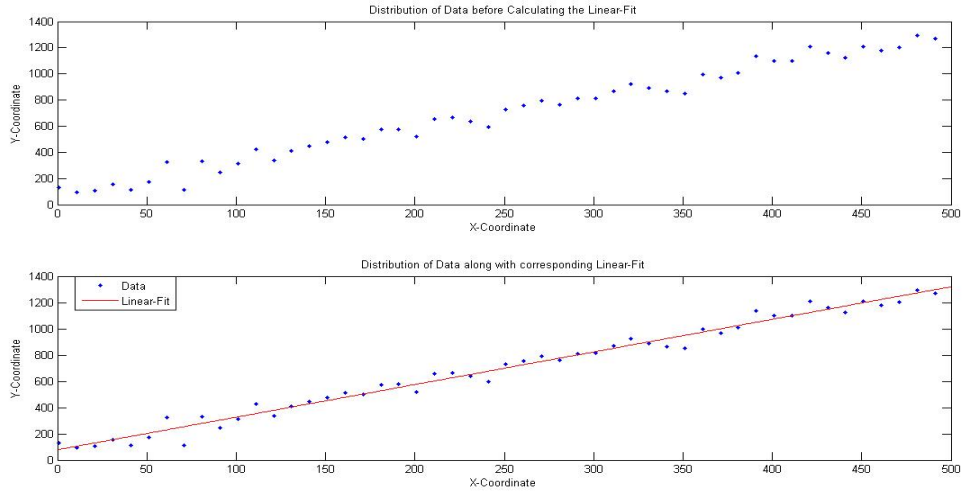

Figure A.1: The distribution of sample data (upper subplot) along with the corresponding least squares linear-fit to the data (lower subplot).

A.2 Least-Square Regression Decomposition (LSRD)

Considering the robots as data points, the data consists of (x, y) pairs of measurements that describe the respective locations of the robots in the environment. Here we assume x and y coordinates as independent and dependent variables, respectively. In other words, higher priority is given to the x-coordinate information of the agents.[1]

Given the distribution of robots, the mission is decomposed into a set of virtual goals whose location information at every decision cycle is collinear with respect to one another. Having such collinearity among virtual goals is represented by the linear equation:

$$Y = aX + b \tag{A.1}$$

We attempt to find the values of two parameters, namely a (the slope of the line) and b (the line intercept with the y-coordinate), so as to minimize the discrepancy between the measured and calculated values y_i and $y(x_i)$ in which y_i is the current y-coordinate of the

[1]This correlation of the x and the y values is not a restricted assumption. Their role can be interchanged. However, this results to a fitted-line that intercepts the x-coordinate (see Figure A.1).

i^{th} robot and $y(x_i)$=ax_i+b represents the approximate functional relationship between x_i and $y(x_i)$, i=1 $\ldots n$. In other words, $y(x_i)$ is the new y-coordinate of the i^{th} robot with respect to collinear virtual goals $\rho_j \in \mathscr{VG}$ in equation (A.1).

Using the Gaussian assumption, for any estimated values of the parameters a and b, the probability of obtaining the observed set of measurements is calculated as (see Bevington and Robinson, 2003):

$$P(x,y) \quad = \quad \Pi_{i=1}^{n}(\frac{1}{\sigma_i\sqrt{2\pi}})e^{(-\frac{1}{2}\sum_{i=1}^{n}[\frac{y_i-y(x_i)}{\sigma_i}]^2)} \tag{A.2}$$

in which quantities $\frac{1}{\sigma_i^2}$ serve as weighting factors. This is to say that the location information of each robot is weighted inversely by its variance σ_i^2. The reason for choosing this criterion as the weighting factor is due to the fact that the distribution information may not have been measured with the same precision. For example, the location of some robots may not be as precise as other robots due to noise or communication failure. Such discrepancies in information precision can be represented by assuming a population distribution with the same mean value μ but different standard deviations σ_i. The goal of finding the optimum fit to robot locations is then expressed by finding a and b values in equation (A.1) that minimize the exponential term (i.e., goodness-of-fit parameter or χ^2) in equation (A.2). In other words, obtaining the smallest sum of the squares or least-squares fit.

Setting the partial derivatives of the exponential term in equation (A.2) (with respect to parameters a and b) to zero, yields (see Bevington and Robinson, 2003):

$$\sum_{i=1}^{n}\frac{y_i}{\sigma_i^2} = a\sum_{i=1}^{n}\frac{1}{\sigma_i^2} + b\sum_{i=1}^{n}\frac{x_i}{\sigma_i^2} \tag{A.3}$$

$$\sum_{i=1}^{n}\frac{x_iy_i}{\sigma_i^2} = a\sum_{i=1}^{n}\frac{x_i}{\sigma_i^2} + \sum_{i=1}^{n}\frac{x_i^2}{\sigma_i^2} \tag{A.4}$$

Solving equations (A.3) and (A.4) for parameters a and b, we have:

$$a = \frac{1}{\Delta}(\sum_{i=1}^{n}\frac{x_i^2}{\sigma_i^2}\sum_{i=1}^{n}\frac{y_i^2}{\sigma_i^2} - \sum_{i=1}^{n}\frac{x_i}{\sigma_i^2}\sum_{i=1}^{n}\frac{x_iy_i}{\sigma_i^2}) \tag{A.5}$$

$$b = \frac{1}{\Delta}(\sum_{i=1}^{n}\frac{1}{\sigma_i^2}\sum_{i=1}^{n}\frac{x_iy_i}{\sigma_i^2} - \sum_{i=1}^{n}\frac{x_i}{\sigma_i^2}\sum_{i=1}^{n}\frac{y_i}{\sigma_i^2}) \tag{A.6}$$

$$\Delta = \sum_{i=1}^{n} \frac{1}{\sigma_i^2} \sum_{i=1}^{n} \frac{x_i^2}{\sigma_i^2} - [\sum_{i=1}^{n} \frac{x_i}{\sigma_i^2}]^2 \tag{A.7}$$

Once the a and b parameters are known, they are used to determine the virtual goal $\rho_j \in \mathscr{VG}$ locations, given the location information of the robots at every decision cycle using:

$$y_i = ax_i + b, \quad i{=}1 \ldots n \tag{A.8}$$

where n is total number of robotic agents.

A.2.1 Effect of the Cost of Relocation of the Robotic Agents

Equation (A.2) interprets the value $\frac{1}{\sigma_i^2}$ as the weighting factor associated with the location information of the robotic agents. We consider this weighting factor to represent the cost of the relocation of the robotic agent from its current location to the location of the allocated virtual goal. The cost of the relocation of the robotic agent is calculated using different weighting criteria. We adapt the following criteria to calculate the cost of the relocation of the robotic agents (see Neter et atl., (1990); Chatterjee and Price, (1991); Kutner et al., (2005) for details):

1. The cost of the relocation of a robotic agent is equal to the inverse of the variance of the distribution of the robotic agents $\frac{1}{\sigma_i^2}$.[2]

2. The cost of the relocation of a robotic agent is equal to the inverse of the square root of the x-coordinate of the robotic agent $\frac{1}{\sqrt{x_i}}$.

3. The cost of the relocation of a robotic agent is equal to the inverse of the y-coordinate of the robotic agent $\frac{1}{y_i}$.

4. The cost of the relocation of a robotic agent is equal to the ratio of the y-coordinate to the x-coordinate of the robotic agent $\frac{y_i}{x_i}$.

[2]The variance of the distribution of the robotic agents is computed as:

$$\sigma_i^2 = \frac{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}{n-1}$$

where y_i, $\bar{y}_i$, and n represent the y-coordinate, the mean value of the y-coordinate, and the total number of the robotic agents, respectively.

5. The cost of the relocation of every agent is 1. This modifies equation (A.5) through equation (A.7) to:

$$\Delta = n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2 \tag{A.9}$$

$$a = \frac{1}{\Delta}(n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i) \tag{A.10}$$

$$b = \frac{1}{\Delta}(\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i) \tag{A.11}$$

Appendix B

Extension to the Isogonic Decomposition

We present an approach to generate a set of virtual goals using the first isogonic point of an isosceles triangle in section 2.3. Furthermore, we demonstrate that to confine the location of the isogonic point within the convex hull of the triangle it is necessary to introduce the upper bound $\angle\rho_2\rho_1\rho_3 < 120^\circ$ (see section 2.3, Figure 2.7 and Theorem 3). This setting guarantees that the location of the isogonic point of the triangle $\triangle\rho_2\rho_1\rho_3$ is confined within the convex hull of its vertices (see section 2.3, Corollary 3). However, it is possible to alleviate this upper bound through the utilization of the second isogonic point of the triangle, if the confinement of the ρ_4 is not necessary.

B.1 Second Isogonic Decomposition

In section 2.3, we state that the first isogonic point of the triangle is always on the intersection of the lines that connect the vertices of the three equilateral triangles formed out of the sides of the given triangle to the vertex that is in the opposite side of the given triangle. We draw these equilateral triangles inward to obtain the second isogonic point of the isosceles triangle. Figure B.1 depicts this scenario. It is apparent that the result of the Theorem 2 holds for the second isogonic point if we draw $\triangle\rho_2 s'\rho_3$ upward and increase $\lambda = \angle\rho_2\rho_1\rho_3 \geq 120^\circ$ in Figure 2.7. The location information of ρ_4 with regards to ρ_1 and

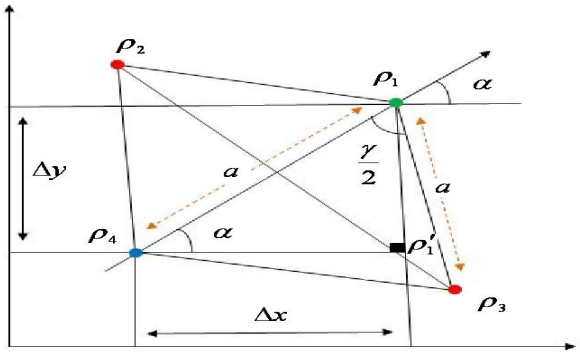

Figure B.1: The second isogonic formation of the virtual goals.

ρ_3[1] when $\angle\rho_2\rho_1\rho_3 \geq 120°$ is calculated as:

$$\rho_4 = \begin{bmatrix} x_{\rho_1} - \triangle x \\ y_{\rho_1} - \triangle y \end{bmatrix} = \begin{bmatrix} x_{\rho_1} - \|\rho_3\rho_1\| \cos(\alpha) \\ y_{\rho_1} - \|\rho_3\rho_1\| \sin(\alpha) \end{bmatrix} \tag{B.1}$$

where:

$$\sin(\alpha) = \frac{\|\rho_1\rho_1'\|}{\|\rho_4\rho_1\|} = \frac{\triangle y}{\|\rho_3\rho_1\|} \Rightarrow \triangle y = \|\rho_3\rho_1\| \sin(\alpha) \tag{B.2}$$

$$\cos(\alpha) = \frac{\|\rho_4\rho_1'\|}{\|\rho_4\rho_1'\|} = \frac{\triangle x}{\|\rho_3\rho_1\|} \Rightarrow \triangle x = \|\rho_3\rho_1\| \cos(\alpha) \tag{B.3}$$

in the right angle triangle $\triangle\rho_1\rho_1'\rho_4$.

The location of ρ_4 is updated in response to the rotation of the configuration of the virtual goals using the transformational matrix in equation (2.51) and the angle of the rotation θ. Therefore, equation (2.53) is changed to:

$$\begin{pmatrix} cos(\theta) \times (x_{\rho_1} - \|\rho_3\rho_1\| \times cos(\alpha)) + sin(\theta) \times (y_{\rho_1} - \|\rho_3\rho_1\| \times sin(\alpha)) \\ cos(\theta) \times (y_{\rho_1} - \|\rho_3\rho_1\| \times sin(\alpha)) - sin(\theta) \times (x_{\rho_1} - \|\rho_3\rho_1\| \times cos(\alpha)) \\ 1 \end{pmatrix} \tag{B.4}$$

Theorem 8. *The first and the second isogonic decompositions produce two sets of virtual*

[1] $\triangle\rho_2\rho_1\rho_3$ is an isosceles triangle, hence $\|\rho_3\rho_1\| = \|\rho_2\rho_1\|$. Furthermore, location infomation of ρ_2 and ρ_3 are calculated using equation (2.39) through equation (2.42), and equation (2.52).

goals $\mathscr{VG}_1$ and $\mathscr{VG}_2$ that are complete, isomorphic graphs.

Proof. 1. Complete Graphs: Let $\mathscr{VG}_j,\ j = 1, 2$ represent the sets of virtual goals for the first and the second isogonic decompositions of Figure 2.7 and Figure B.1. The completeness of the $\mathscr{VG}_1$ and the $\mathscr{VG}_2$ follows from the observation that every $\rho_i \in \mathscr{VG}_j$ is connected to the remaining $n-1$ virtual goals such that:

$$\forall(\rho_p, \rho_q) \in \mathscr{VG}_j\ p \neq q, \exists e \in E(\mathscr{VG}_j),\ j = 1, 2 \tag{B.5}$$

2. Isomorphism: Since $\mathscr{VG}_1$ and $\mathscr{VG}_2$ are both complete graphs of the same number of virtual goals, it follows that any pairing off of virtual goals gives a corresponding pairing of the edges and are isomorphic to each other.

□

Corollary 4. *Isogonic mission decomposition results in a unique, shortest connectivity link among virtual goals $\rho_j \in \mathscr{VG}$.*

Proof. Let ρ_4 represent the isogonic point of the isosceles triangles in Figure 2.7 and Figure B.1. This implies that ρ_4 is the point that minimizes the cumulative sum of Euclidean distances of the vertices of $\triangle\rho_2\rho_1\rho_3$. Therefore, ρ_4 satisfies:

$$\min \sum_{i=1}^{3} w_i \|\rho_i - \rho_4\| \tag{B.6}$$

□

B.2 Extendability

We demonstrate that the initial location information of one of the virtual goals is sufficient to decompose a mission into a set of virtual goals $\mathscr{VG}$ in section 2.3 and section 2.3.1. However, we limit the scope of this procedure to a special case where the cardinality of a set of virtual goals is four. We extend the formulation of the isogonic decomposition to generate an arbitrary number of virtual goals in this section. More specifically, we show that the growth of the cardinality of a set of virtual goals is theoretically not limited when the initial location information of one of the virtual goals is known.

We consider $\rho_1 \in \mathscr{VG}$ to represent the virtual goal with the known location information.

Furthermore, we use the term *cluster* to refer to a set of virtual goals that is generated using the isogonic decomposition formalism in section 2.3. We define a cluster as follows.

Definition 12 (Virtual Goals Cluster)**.** *Cluster $\mathscr{C}_i$ of a set of virtual goals is a subset of the $\mathscr{VG}$ that satisfies Theorem 8 and Corollary 4.*

It is apparent that $\forall \mathscr{C}_i$:

$$\mathscr{C}_i \subseteq \mathscr{VG} \tag{B.7}$$

$$\bigcup_{i=1}^{c} \mathscr{C}_i = \mathscr{VG} \tag{B.8}$$

$$\bigcap_{i=1}^{c} \mathscr{C}_i = \emptyset \tag{B.9}$$

where c is the cardinality of the cluster $\mathscr{C}_i$. Equation (B.7) through equation (B.9) express that a cluster is a subset of the set of virtual goals that satisfies Definition 1 (see Chapter 2).

Figure B.2 illustrates the concept of the clusters of virtual goals using the isogonic decomposition formalism. $\rho_1 \in \mathscr{C}_1$ is the virtual goal with the known initial location information. The arrows show the direction of the generation of virtual goals based on $\rho_1 \in \mathscr{C}_1$.

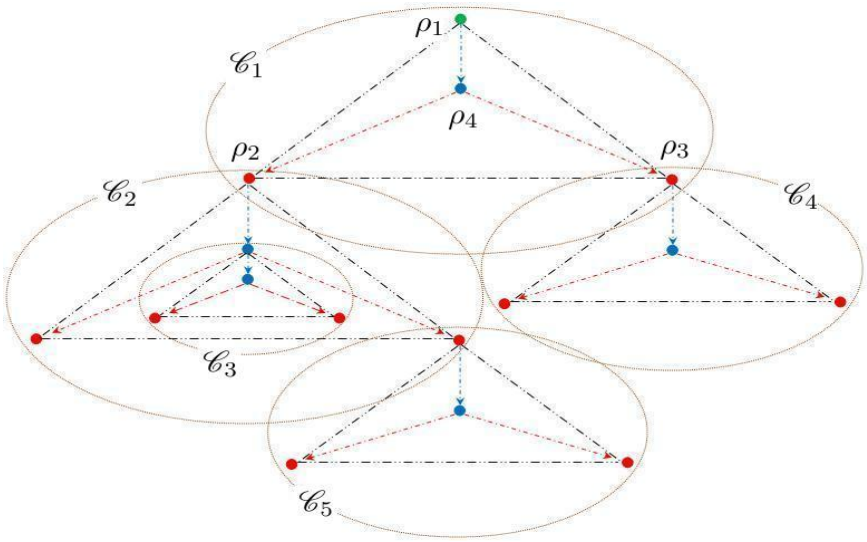

Figure B.2: Isogonic decomposition extendability. Clusters are labeled $\mathscr{C}_i$, $i = 1 \ldots 5$. Arrows show the flow of the calculation of the subsequent clusters using $\rho_1 \in \mathscr{C}_1$.

Proposition 1. *Isogonic decomposition generates an arbitrary number of virtual goals if the initial location information of one of the virtual goal is known.*

Proof. Let ρ_1 represent the virtual goal with known initial location information. It is possible to generate the remaining virtual goals $\rho_i,\ i = 2, 3, 4$ using ρ_1 and the isogonic decomposition formulation. These virtual goals form the cluster $\mathscr{C}_1$. It is apparent that $\mathscr{C}_1$ satisfies Definition 1. Furthermore, $\rho_2,\ \rho_3 \in \mathscr{C}_1$ are the virtual goals of the clusters $\mathscr{C}_2$ and $\mathscr{C}_3$ (see Figure B.2) where the initial location information of the virtual goal is known. Therefore, $\rho_i \in \mathscr{C}_j,\ j = 2, 3$ are calculated based on $\rho_1 \in \mathscr{C}_1$. This procedure can be extended up to any arbitrary number of steps to generate a new cluster using the location information of $\rho_1 \in \mathscr{C}_1$.

1. Every cluster is generated using the isogonic decomposition formalism. This implies that $\forall \mathscr{C}_i \subseteq \mathscr{VG},\ \exists \rho_i \in \mathscr{C}_i$ where this virtual goal corresponds to the isogonic point of the cluster (i.e., the blue-colored nodes in Figure B.2). Therefore, every cluster $\mathscr{C}_i$ satisfies Theorem 8 and Corollary 4.

2. Every cluster is generated using the location information of the virtual goal of the succeeding cluster. For instance, $\mathscr{C}_2$ is calculated using $\rho_2 \in \mathscr{C}_1$ that is calculated based on $\rho_1 \in \mathscr{C}_1$. Furthermore, every cluster satisfies the Definition 1 (see case 1 of the proposition 1). Therefore, it is possible to reduce a cluster to its leading virtual goal (e.g., ρ_2 is the leading virtual goal of $\mathscr{C}_2$) with respect to its succeeding cluster. This results in every proceeding cluster to be a virtual goal of its succeeding cluster that satisfies Theorem 8 and Corollary 4. This follows through the observation that the leading virtual goal of every proceeding cluster is also a virtual goal of the succeeding cluster that satisfies the Definition 1.

□

Appendix C

Internal State Component

The internal state component $\phi_i(r_i, \rho_j)$ ranks a set of virtual goals $\mathscr{VG}$ based on the amount of the energy that is available to the robotic agents. This component utilizes the location information of the virtual goals $\rho_j \in \mathscr{VG}$ along with the energy information to determine if the agents are able to reach these virtual goals:[1]

$$\phi_i(r_i, \rho_j) = \pi_i^t(e_i \mapsto \rho_j) = \begin{bmatrix} 0 & e_i \leq \mathscr{EN}_i \\ \frac{e_i - \mathscr{EN}_i}{\mathscr{FE}_i} & Otherwise \end{bmatrix} \tag{C.1}$$

where

e_i: current energy level of the i^{th} robotic agent.

$\mathscr{FE}_i$: energy available to the i^{th} robotic agent when it is fully charged (i.e., Full Energy).

$\mathscr{EN}_i$: energy needed by the i^{th} robotic agent to reach $\rho_j \in \mathscr{VG}$.

Equation (C.1) verifies the necessity of the multiplicative incorporation of the internal and the external states of the decision engine in equation (3.1). In particular, this equation expresses that the result of the computation of the internal state component is zero if the current energy level e_i of the agent is less than energy needed to reach a virtual goal ρ_j. As a result, the final vote of the agent $\pi_i(\rho_j)$ is zero.

[1] Although it is possible to model the internal state component using the formal system of propositional logic, a numerical representation of the ranking is more desirable. This is due to the fact that the rankings of the virtual goals that are represented quantitatively provide the system with better estimates of the final vote values at every decision cycle.

C.1 Computation of the Energy Consumption

Mei et al. (2005) formulate the power consumption of a robotic agent as the cumulative sum of the mechanical power output and the transformation loss. Let m_i, a_i, and v_i represent the mass, the acceleration, and the velocity of a robotic agent. The traction force needed by the agent to traverse a given path is calculated as:

$$m_i \times (a_i + g \times \mu) \tag{C.2}$$

where g and μ represent the gravitational and the ground friction constants. Therefore, the output mechanical power is formulated as:

$$m_i(a_i + g \times \mu) \times v_i \tag{C.3}$$

Using equations (C.2) and (C.3) the motion power of the robotic agents is calculated as:

$$p_{m_i}(v_i) = p_l + m_i(a_i + g \times \mu) \times v_i \tag{C.4}$$

where p_l is the transforming loss of the i^{th} robotic agent.

We use the following energy model during the simulation (see Mei et al. (2005) for further explanation):

$$p_{m_i}(v_i) = 0.29 + 7.4 \times v_i \tag{C.5}$$

where the mass of a robotic agent is $m_i = 9kg$.

C.1.1 Effect of the Internal State Component

We formulate the external state component and the effect of the opportunistic ranking module of the component in section 3.2.1. The next step to finalize the vote profile of the robotic agents (see Chapter 3, Definition 5) is the calculation of the estimate of the internal state component for a set of virtual goals. These components are incorporated multiplicatively to rank the virtual goals at every decision cycle.

Let Table C.1 represent the estimates of the external state component of the robotic agent r_1 for a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$. Let assume the battery of the robotic

Table C.1: The estimate of the external state component of agent r_1 for a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ at decision cycle $t+1$

r_1	ρ_1	ρ_2	ρ_3
$\psi_1(r_1, \rho_j)$	0.4925	0.0	0.5075

agent r_1 holds 10000 *Joules* when it is fully charged and that r_1 current energy level is 9000 *Joules*. Let further assume that the current distance of the robotic agent r_1 from the virtual goal ρ_1 is 37.76 m, and that the velocity of the robotic agent r_1 has the upper bound of $v_1 = 20\ \frac{m}{s}$. The estimate of the internal state component of r_1 for the virtual goal ρ_1 using equations (C.5) and (C.1) is:

$$\phi_i(r_1, \rho_1) = \frac{9000 - [(0.29 + 7.4 \times 20) \times 37.76]}{10000} = 0.34 \tag{C.6}$$

We similarly compute the estimate of the internal state component of r_1 for all available virtual goals using its current distances to the virtual goals.

Let Table C.2 represent the estimate of the internal state component of r_1 for the virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$. The entries of Table C.2 indicates that the internal state component of r_1 ranks the virtual goal ρ_1 with the highest estimate of success. This estimate apparently contradicts the estimate of the external state component of r_1 that ranks ρ_3 with the highest estimate. However, the multiplicative incorporation of the external and the internal states of r_1 provides the agent with better estimate of success for the available virtual goals.

We use the entries of Table C.1 and Table C.2 along with equation (3.1) to finalize the vote profile of r_1:

$$\pi_1(\rho_1) = 0.4925 \times 0.34 = 0.167450$$
$$\pi_1(\rho_2) = 0.0 \times 0.33 = 0.0$$
$$\pi_1(\rho_3) = 0.5075 \times 0.33 = 0.167475$$

Table C.2: The estimate of the internal state component of agent r_1 for a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ at decision cycle $t+1$

r_1	ρ_1	ρ_2	ρ_3
$\phi_1(r_1, \rho_j)$	0.34	0.33	0.33

Furthermore, we normalize these vote values (see section 3.2.1 for further explanation) to obtain a vote profile that satisfies equation (3.2). Table C.3 shows the final vote profile of r_1 after the normalization of the multiplicative incorporation of the internal and the external states of the agent. The entries of Table C.3 indicates that the decision mechanism of r_1 ranks the virtual goal ρ_3 with a slightly higher estimate of success. However, the difference between the vote values of the virtual goals ρ_1 and ρ_3 is negligible.

Table C.3: The vote profile Π_1 of agent r_1 for a set of virtual goals $\mathscr{VG} = \{\rho_1, \rho_2, \rho_3\}$ after multiplicative incorporation of the internal and the external states components at decision cycle $t + 1$.

r_1	ρ_1	ρ_2	ρ_3
$\pi_1(\rho_j)$	0.49998	0.0	0.50002

Appendix D

Case Study Algorithms

We provide a brief description of the algorithms that are used in Chapter 5 and Chapter 7. They are the Brownian motion (see section 5.1), the Hungarian algorithm (see section 5.2), and the Bayes filter (see section 7.3).

D.1 Brownian Motion

The Brownian motion models the emergence of the motion of the particles that are moving randomly in d-dimensional space. Such a motion is due to a random displacement of the particles where every displacement is considered to be within a short range. In other words, the displacements do not exhibit big jumps. Collisions among the particles or an externally exerted force are the examples of the cause of the displacement of the particles.

If S_0 represents the position of a particle at time zero, the displacement of this particle at time n is formulated as (Morters and Peres, 2010):

$$S_n = S_0 + \sum_{i=1}^{n} X_i \tag{D.1}$$

where $X_1 \dots X_n$ are the displacements that are assumed to be independent, identically distributed random variables with values in $\mathbb{R}^d$. Hence, the process $\{S_n : n \geq 0\}$ represents a random walk where the displacements are the inputs to this process. Morters and Peres (2010) state that all random walks that are derived from the displacements with the same mean and covariance matrix generate the same Brownian motion. This observation substantially relaxes the assumption of the independence of the identically distributed displacements

to generate the motion of the particles.

D.1.1 Simulation of Brownian Motion

We generate the Brownian motion of the subtasks in Chapter 5 using the following Matlab code snippet. It shows the generation of the motion for a set of 20 data points. The location information of these data points is generated by $randn(n, 1)$ function that returns an n-by-1 matrix of pseudo-random values that are drawn from the standard normal distribution.[1] $s = 0.03$ is the drifting parameter that defines the rate of displacement of the data points at every iteration.[2]

```
n = 20 //Total number of data points
s = .03
x = rand(n,1)-0.5; //Randomly generated x-coordinate values
y = rand(n,1)-0.5; //Randomly generated y-coordinate values
h = plot(x,y,'.'); // plot these data as dots
while 1
   drawnow //update the relocations of the data points on the screen
   x = x + s * randn(n,1);
   y = y + s * randn(n,1);
   set(h,'XData',x,'YData',y) // update x- and y-coordinate values
end
```

D.2 Hungarian Algorithm

The Hungarian algorithm uses the complete bipartite graph to model the multi-robot, multi-task allocation problem. It divides the agents and the task space into two disjoint sets of vertices where every vertex of the *agent set* is connected to every other vertices of the *task set* and vice-versa. As a result, the generated bipartite graph forms an undirected graph. This bipartite graph is utilized to calculate an optimal allocation of the tasks to the robotic agents where every agent is assigned to a task. The allocated tasks are distinct. Therefore,

[1] We introduce the range of these data to reside within the environment during the simulation.

[2] This drifting parameter is denoted by μ (see Morters and Peres, 2010). We use the drifting parameter $\mu = 0.03$ throughout the simulations.

it requires the number of agents and the cardinality of the task space to be the same.

The Hungarian algorithm is summarized in Algorithm 7. It matches the agents and the tasks where the result is an optimal allocation. $N(u)$ indicates the neighboring vertex of a given vertex u. S and T contain the final matching of the agents and the tasks. X and Y are the *agent set* and the *task set*, respectively.[3]

Algorithm 7: Hungarian algorithm.

Data: An $n \times n$ that represents a complete wighted bipartite graph $G = \{X, Y, E\}$ where $|X| = |Y| = n$.

```
1  begin
2      Generate an initial labeling l and matching M in G;
       Pick a random vertex u ∈ X. Set S = {u}, T = ∅;
       if N(u) = T then
3          δ = min_{x∈S, y∈Y−T} {l(x) + l(y) − w(x, y)};
           if v ∈ S then
4              l′(v) = l(v) − δ;
5          else if v ∈ T then
6              l′(v) = l(v) + δ;
7          else
8              l′(v) = l(v);
9      if N(S) ≠ T then
10         Pick y ∈ N(S) − T;
           if y is available then
11             Augment u → y path to M. Go to step 2;
12         else if y matched z ∈ T then
13             S = S ∪ {z}, T = T ∪ {y}, Go to step 3;
```

D.3 Bayes Filter

The Bayes filter utilizes the measurement (e.g., distance, sensor reading) and the control data (e.g., velocity, actuator, end-effector) to calculate the belief distribution of an agent in conjunction with a given task (e.g., ranking different virtual goals). It is a recursive algorithm where the belief at time t is calculated based on its value at time $t-1$. Algorithm 8 shows a single iteration of the Bayes filter. The belief of the agent at $t-1$ along with its current control u_t and measurement z_t are the inputs to the algorithm.[4]

[3] See (Liu and Shell, 2011, p. 938) for an illustrative example.

[4] z_t is analogous to the default ranking module of the decision engine (see equation 3.4). Although we do not explicitly include the control data in the decision engine (see equation3.1), it is similar to the reactive

Algorithm 8: Bayes filter.

Data: z_t, The measurement at time t.
Data: u_t, The control data at time t.
Data: $bel(x_{t-1})$, Belief calculated at time $t-1$.

```
1 begin
2     for x_t do
3         bel(x_t) = ∫ p(x_t|u_t, x_{t-1}) bel(x_{t-1}) d_{x_{t-1}};
          bel(x_t) = η p(z_t|x_t) bel(x_t);
4     return bel(x_t);
```

In line 3, the algorithm processes the control u_t by calculating the belief of the agent over the sate x_t using the prior $bel(x_{t-1})$. This is done through the integration (i.e., sum) of the product of the prior value assigned to x_{t-1} and the probability of the transition from x_{t-1} to x_t induced by u_t. It then multiplies this result by the probability that indicates the measurement z_t has occurred.[5] It is possible that the result of the multiplication does not integrate to 1. Hence, it is normalized by η to obtain the final belief of the agent at $bel(x_t)$.[6]

control of the agents to calculate their votes during the collision avoidance (see equation 3.6).

[5]This probability and the prior assigned to x_{t-1} form the *a priori* information of this framework.

[6]see (Thrun et al., 2006, p. 29; Keshmiri and Payandeh, 2011) for numerical examples.