AN INTELLIGENT AND UNIFIED FRAMEWORK FOR MULTIPLE ROBOT AND HUMAN COALITION FORMATION

By

Sayan D. Sen

Dissertation
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

May, 2015

Nashville, Tenessee

ACKNOWLEDGMENTS

"Take up one idea. Make that one idea your life - think of it, dream of it, live on that idea. Let the brain, muscles, nerves, every part of your body, be full of that idea, and just leave every other idea alone. This is the way to success, that is way great spiritual giants are produced."

-Swami Vivekananda

I have been privileged to be under the guardianship of my research adviser, Dr. Julie A. Adams, who has always pushed me to challenge myself and bring the best out of me. I cannot thank her enough for the immense support and encouragement that she provided for my new ideas and endeavors during the entire length of my research. I am obliged to Dr. Douglas Fisher and Dr. Gautam Biswas for fruitful insights into various research topics. I also thank Dr. Nilanjan Sarkar and Dr. Peter Stone for valuable suggestions and discussions that helped me to pursue my PhD. I am grateful to Sean for providing insights into statistical analysis and Electa for her help with understanding the Human-Machine Teaming Laboratory's graphical user interface. This dissertation research was partially supported by an ONR DEPSCOR Award# N000140911161.

I take this humble opportunity to thank my amazing parents and sister whose love and belief in me have propelled me through thick and thin of this thing called *"life"*. I particularly thank my wife, Deblina for all her support, love, and motivation throughout this entire journey. Her words of encouragement and continuous instilling of faith in myself emboldened my spirits during the downs of my life. I thank Almighty God for showering her blessings for all that I have achieved in life.

Finally, I thank Nihaar, Indranil, Akash, Shubhajit, and Debargah for being such good friends and making my stay in Nashville exhilarating.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

Introduction

I.1 Problem Statement

Robotic systems have proven effective with recent deployments of unmanned robots in a number of mission situations (e.g., search and rescue, first response, space) (Liu and Nejat, 2013; Pedersen et al., 2003). Moreover, unmanned systems are increasingly leveraged in the military domain for diverse applications (e.g., surveillance, reconnaissance) (Sparrow, 2009). Complex mission requirements are often greater than the capabilities of a single agent (human or robotic asset); therefore, multi-agent coalition formation, which intelligently groups agents, is necessary to perform tasks collectively. Exploiting the diverse capabilities of heterogeneous members in such coalitions can result in a significant improvement in the team's performance when executing a mission.

Teaming multiple agents requires efficient coalition formation techniques. Coalition formation is an *NP*-complete problem (Sandholm et al., 1999) that is also hard to approximate within a reasonable factor (Service and Adams, 2011a). The computational complexity of the problem, owing to an exponential search space has led to the development of a number of greedy algorithms (Shehory and Kraus, 1998; Vig and Adams, 2006b), approximation approaches (Dang and Jennings, 2004a; Rahwan et al., 2009a; Sandholm et al., 1999), and market-based techniques (Dias, 2004; Shiroma and Campos, 2009; Vig and Adams, 2006a); however, all these categories of algorithms have their respective drawbacks. Greedy algorithms generate solutions quickly, but do not guarantee solution quality and often result in suboptimal solutions. Conversely, approximation algorithms guarantee solution quality, but are rendered inapplicable for real-time mission scenarios due to their exorbitant worst-case run-time complexity ($O(n^n)$ for anytime and $O(3^n)$ for design-to-time algorithms). Market-based approaches provide a robust and fault-tolerant distributed infrastructure, but have high communication bandwidth requirements. Therefore, a single class of algorithms will not be robust and flexible enough to handle a wide spectrum of highly dynamic real-world missions. Greedy algorithms leverage some heuristics that only perform well when the relevant information is available. The dynamics and uncertainties of real-world environments do not guarantee the availability of such information in every mission scenario; therefore, rendering a particular heuristic-based algorithm less applicable. Heuristic algorithms have been shown experimentally to be effective only in compatible real-world missions (DeJong, 2005).

The primary contribution of this dissertation is the development of a state-of-the-art unified framework, called the **i**ntelligent-**C**oalition **F**ormation framework for **H**umans **a**nd **R**obots (*i-CiFHaR*), the first of its kind, which incorporates a library of diverse coalition formation algorithms, each employing a different problem solving mechanism. The framework employs unsupervised learning to mine crucial patterns and intricate relationships among the algorithms that result in important clusters of algorithms. Equipped with Bayesian reasoning, *i-CiFHaR* makes intelligent and optimized decisions over the library and selects the most appropriate algorithm(s) to apply in accordance with multiple mission criteria and environmental constraints in order to generate task coalitions. The library of diverse algorithms, coupled with the intelligent algorithm selection capability enables *i-CiFHaR* to generate robust solutions and handle contingencies in a wide variety of dynamic, real-world missions; thereby, rendering it as an effective decision support system for mission supervisors.

Real-world missions may have a greater propensity towards using greedy algorithms, which can generate solutions of acceptable quality within the stipulated time. A commonly leveraged heuristic in many existing heuristic-based coalition formation algorithms is one that constrains the coalition size up to a maximum limit, k (Shehory and Kraus, 1998; Tošić and Agha, 2005; Vig and Adams, 2006b). Despite this restriction, the coalition formation problem remains *NP*-complete and the algorithms are rendered inapplicable when the task requirements exceed the collective capabilities of all coalitions up to size, k. There exists a void in the field of multi-agent systems for heuristic algorithms that can scale well for large teams of agents (robots and human assets) without the use of such limited heuristics. The second major contribution of this dissertation fills this gap by presenting two hybrid **s**imulated **A**nnealing-inspired **ANT** colony optimization algorithms, called the *sA-ANT* and the *sA-ANT** that are applicable to a wide spectrum of combinatorial optimization problems, including the coalition formation problem.

Ant colony optimization (ACO) algorithms are swarm-based, biologically inspired search algorithms that simulate the collective foraging behavior commonly exhibited by biological swarms of ants. Real ants, while searching for food use an indirect, local, and environment-based communication mode in order to exchange information through the use of pheromones. However, ACO approaches suffer from two major drawbacks: (1) search stagnation, which often results in sub-optimal solutions, and (2) higher computational time. The *sA-ANT* and *sA-ANT** algorithms are important contributions to the swarm intelligence community, because they address the search stagnation drawback by introducing two novel pheromone update policies that incorporate the simulated annealing methodology. These pheromone update policies deviate significantly from existing state-of-the-art ACO algorithms that employ varying strategies to deposit pheromones including: using a set of elitist ants, *Elitist Ant System* (Dorigo et al., 2006); leveraging solution quality-based ranked ants, *Rank-Based Ant System* (Bullnheimer et al., 1997); depositing pheromones using the single best ant, *Ant Colony System* (Dorigo and Gambardella, 1997); and bounding of the pheromone levels, *Max-Min Ant System* (Stutzle and Hoos, 2000). *sA-ANT* dynamically modulates the number of ants responsible for depositing pheromones at the end of each iteration, while *sA-ANT** stochastically selects the best fit solution from a repository of good solutions to deposit pheromones. Both depositing policies result in the algorithms' enhanced search exploration in the initial phases, and improved exploitation during the later stages. The high initial annealing temperature, coupled with a gradual annealing schedule prohibits the algorithms from stagnating in local optima.

The effectiveness of the two variants of *sA-ANT* as generic search algorithms was demonstrated by applying them to the Traveling Salesman Problem (TSP), the maximal clique problem, and the multi-agent coalition formation problem. The performance of *sA-ANT** and *sA-ANT* for the TSP is very promising, because it generates significantly shorter tour lengths for several TSP instances, when benchmarked against existing state-of-the-art ACO approaches. Furthermore, the potential of the developed swarm algorithms as an efficient greedy coalition formation algorithm for large teams of agents up to a size of 200 has also been successfully demonstrated. Considering real-world applications, the algorithms when applied to the coalition formation problem optimize the robots' traveling cost to the assigned task, communication cost among coalition members, and the loss of resources (e.g., battery, camera). Therefore, this dissertation provides a swarm-based algorithm that contributes to the multi-agent systems community as an effective greedy coalition formation algorithm that computes high quality coalitions for very large teams of robots and humans without constraining coalition sizes. Aiming to address the drawbacks of the centralized algorithms that are inherently brittle and unresponsive to dynamic environments, this dissertation also presents the distributed variant of *sA-ANT**, which creates robust coalitions of heterogeneous robots for real-world missions. The decentralized algorithm, *d-sA-ANT** employs an information propagation mechanism that facilitates information sharing among the robots based on their immediate local neighborhood. Experimental results demonstrate that the distributed algorithm generates coalitions of virtually identical quality as that of its centralized version, but requiring a significantly higher computational time.

A mission supervisor's performance, efficiency, and awareness are adversely affected during mission deployments that cause immense stress and fatigue, which can detrimentally affect his/her decision making ability, thus jeopardizing mission success. Therefore, the ability of *i-CiFHaR* to provide effective decision support to the human supervisor is crucial. The role of *i-CiFHaR*'s decision support is to help the human supervisor make effective decisions during critical missions in uncertain and dynamic environments. The coalition formation algorithm(s) selected by *i-CiFHaR* may not always satisfy all the criteria and constraints provided by the human supervisor. When the framework is unable to select a single best algorithm satisfying all mission requirements, it may select a subset of algorithms suitable for the mission. The decision support feature presents the supervisor with necessary metrics, such as the *expected utility* score of each coalition formed by the selected algorithm(s). Given the ranking of algorithms, the supervisor can either select the algorithm deemed to be most appropriate according to the missions' needs, or provide further mission specifications, so that the system can revise and refine its algorithm selection.

An important part of a decision support system for coalition formation is an efficient interaction tool between the human mission supervisor and the deployed human-robot teams. A decision support system interface is indispensable for deployed missions for two main reasons. First, it facilitates the transfer of mission specifications, constraints, and criteria to the available team. Second, the interface enables the human supervisor to view crucial information about the progress of tasks (e.g., ongoing, standby, completed), status of the deployed agents (e.g., idle, engaged, faulty), discovery of new tasks, task-coalition allocations, and other contingencies. This dissertation integrates *i-CiFHaR* with the Human-Machine Teaming Laboratory's human-robot interface to further enhance its effectiveness.

I.1.1 Dissertation Roadmap

The subsequent chapters are organized as follows. Chapter II provides a background covering research in the fields of multi-agent coalition formation and ant colony optimization. Chapter III presents the two variants of the hybrid ant colony optimization algorithms, namely the *sA-ANT* and the *sA-ANT**. The chapter discusses the effectiveness of *sA-ANT* and *sA-ANT** in addressing the search stagnation drawback of contemporary ACO approaches. The experimental design and results of applying the algorithms to three combinatorial optimization problems are provided and discussed. Furthermore, the chapter presents a distributed variant of the *sA-ANT**, which leverages an information propagation mechanism through real-world communication networks to compute multi-robot coalitions in a decentralized fashion. Chapter IV presents the *i-CiFHaR* framework and discusses each of its components that contributes to the decision making process. The chapter also provides the methodology to incorporate unsupervised learning techniques in order to partition the algorithms in its library, which improves *i-CiFHaR*'s computation time by 67%. *i-CiFHaR* was applied to 24 missions scenarios for evaluating its decision making capability and the results are provided in the chapter. The dissertation concludes in Chapter V, which summarizes the contributions and potential future research direction in the field of multi-agent systems.

CHAPTER II

Literature Review

Developing a multi-agent coalition formation framework to accomplish a set of tasks cooperatively in real-world settings is a challenging problem that requires: (1) effective task allocation, (2) agent coalition formation, (3) team coordination, (4) task execution monitoring, and (5) contingency handling. This chapter summarizes several state-of-the-art multi-agent/robot coalition formation and coordination architectures.

II.1 Multi-agent Coalition Formation

A primary focus of this dissertation is on multi-agent coalition formation, an *NP*-complete problem (Sandholm et al., 1999) that is also hard to approximate within a reasonable factor (Service and Adams, 2011a). Given a set of *n* agents, the exponential number of possible coalitions makes the problem extremely challenging to solve. The multi-agent coalition formation problem is synonymous to the coalition structure generation problem in characteristic function games, where a coalition structure consists of multiple, disjoint partitions (coalitions) of the agents (Sandholm et al., 1999). The objective of the coalition structure generation problem is to realize the best coalition structure that maximizes the total social welfare or utility value.

II.1.1 Problem Formulation

Given a set of agents along with their resources and capabilities, and a set of tasks specified with their requirements and parameters (e.g., task location, priority, dependencies, utility), the coalition formation problem objective is to compute the best fit agent coalition for a given task.

Definition 1. *Let a team of n agents, $A = \{A_1, A_2, ..., A_n\}$ attempt to complete a set of tasks, T that is comprised of m tasks, $T = \{T_1, T_2, ..., T_m\}$, with each task requiring a set of r resource types to be accomplished. Each robot is equipped with a set of r resources (e.g., camera, sonar, laser), denoted by Res^{A_i} or a set of r services (e.g., sentry-duty, box-pushing, mapping), denoted by Ser^{A_i} such that:*

$$\forall A_i \in A, \exists Res^{A_i} = \{Res_j^{A_i}\} \mid 1 \leq i \leq n, 1 \leq j \leq r. \tag{II.1}$$

$$\forall A_i \in A, \exists Ser^{A_i} = \{Ser_j^{A_i}\} \mid 1 \leq i \leq n, 1 \leq j \leq r. \tag{II.2}$$

Definition 2. *A robot coalition, $S_c \subseteq A$ can only accomplish a given task, $T_k \in T$, if and only if the combined* *resource or service capabilities of the team, $S_c^{cap} = \{SR_c^{cap,1}, SR_c^{cap,2}, ..., SR_c^{cap,r}\}$ exceed the task's requirements, i.e., $\forall 1 \leq j \leq r, S_c^{cap,j} \geq Res_j^{T_k}$ or $\forall 1 \leq j \leq r, S_c^{cap,j} \geq Ser_j^{T_k}$.*

The existing coalition formation algorithms can be broadly classified into three major categories: (1) greedy, (2) market-based, and (3) approximation approaches. Given a set of tasks, the coalition formation problem evolves into a special case of the coalition structure generation problem, where the agents are partitioned into disjoint coalitions, each assigned to a specific task. This similarity between the coalition formation problem for task allocation and the generic coalition structure generation problem motivates the discussion of existing approximation algorithms designed for the latter that can be tailored for the coalition formation problem.

II.1.2 Greedy Coalition Formation Algorithms

One of the earliest distributed, heuristic based coalition formation algorithms was proposed for multi-agent task allocation in a Distributed Problem Solving environment (Shehory and Kraus, 1998). The heuristic permitted coalitions upto a maximum size, *k*. Smaller, restricted coalition sizes were justified for multi-robot systems by virtue of lower communication costs and computational complexities. The greedy algorithm used group-rational agents, generated both overlapping and disjoint coalitions, and minimized the overall system cost. The algorithm handled tasks with precedence constraints by employing a separate pre-processing sub-algorithm to allocate only those tasks that had no dependencies at a given time; however, the coalition formation problem still remained NP-hard despite the usage of a maximum upper limit on the coalition sizes.

Inherent differences between multi-agent and multi-robot domains restrict the applicability of Shehory and Kraus' coalition formation algorithm in real-world environments. Addressing this issue, Vig and Adams (2006b) extended the heuristic algorithm (Shehory and Kraus, 1998) to apply to multi-robot domains. The extended algorithm modeled resources as non-transferable owing to the fact that robot resources (e.g., laser, sonar, camera, gripper) are physical entities. Each task was defined as a constraint graph with edges representing the sensor constraints and nodes representing the required sensors and actuators. The improved coalition algorithm utilized arc-consistency to validate each potential task coalition and achieve valid assignments of coalition members to respective sub-tasks. Robust coalitions were calculated for each mission task by introducing a *fault-tolerance coefficient*, which was a function of the *coalition imbalance coefficient* and the coalition size. The degree of unevenness in resource contribution of the coalition members defined the *coalition imbalance coefficient*. The extended algorithm used Shehory and Kraus' heuristic of allowing coalitions of a maximum size, *k* and minimized the total system cost; however, only non-overlapping task coalitions were permitted.

Service and Adams (2011a) proposed a resource/service model based coalition formation algorithm that is very similar to the previous heuristic algorithms (Shehory and Kraus, 1998; Vig and Adams, 2006b) in that it employed the same greedy task allocation procedure and used the same heuristic of limiting coalition sizes; however, it maximizes the system utility, rather than minimizing the total system cost.

A coalition formation algorithm was presented that utilized an underlying organization hierarchy as the heuristic to compute task coalitions in polynomial time (Abdallah and Lesser, 2004). The leaf nodes in the organization represent the robots and the non-leaf nodes represent computational units (managers). Each manager controls its corresponding sub-tree in the organization containing a subset of robots. A task arriving at a particular manager, M is accomplished successfully only if the leaf nodes (robots) in the manager's sub-tree have enough resources. Otherwise, the discovered task is handed over to its parent manager, which decomposes the task into sub-tasks and assigns its children managers to accomplish the task. The managers used Q-learning with neural networks to optimize local decisions within the organization and as a result the coalition algorithm generated improved coalitions over time as each manager gained more experience.

A clique-based, distributed coalition formation algorithm was proposed (Tošić and Agha, 2005) that bounded the sizes of agent coalitions to a maximum upper limit, k and leveraged a topology network that captured inter-agent communications. Agent coalitions of modest sizes were generated that formed maximal cliques in the topology network. The major advantage of this algorithm is its low communication bandwidth requirements, given a sparse topology network.

Weerdt et al. (2007) presented a greedy, distributed coalition formation algorithm for the social network based task allocation problem, a variant of the general task allocation problem. The agent coalitions were computed based on an underlying social network that was derived from inter-agent communication links. A task defined by its required resources, utility, and task location arrives at a particular agent (manager), which recruits its directly connected neighboring agents (contractors) in the network to form coalitions. The task allocation problem, even in the presence of a social network (serving as a heuristic) with node degrees greater than three proved to be a *NP*-complete problem. Therefore, a greedy algorithm was proposed to find coalitions in $O(nm)$ run-time that required $O(n^2m)$ number of communication messages for n tasks and m agents.

Gaston and desJardins (2005) proposed a greedy algorithm also leveraging an agent-oriented network that was based on inter-agent communications to calculate effective coalitions. Each agent formed a node in the network, possessed a single skill set, and acquired one of the three agent states: *Uncommitted*, *Committed*, and *Active*. Tasks were introduced at fixed time intervals and globally advertised with the required skill sets for a fixed time length. The performance of the proposed greedy algorithm was determined by two measures: (1) *global performance*, defined as a ratio of the number of successful task coalitions formed to the total number of tasks introduced, and (2) *local performance* of each agent. The algorithm allowed time-extended task allocations.

A two-stage, distributed coalition formation algorithm specific to unmanned aerial vehicles (UAVs) was proposed that concentrated on: (1) minimizing coalition sizes, and (2) minimizing task completion time (Sujit et al., 2008). The UAV that discovers a given task becomes the task leader and broadcasts the task's resource requirements to all other UAVs, following which the available UAVs respond with their resource contents and traveling costs. Each UAV calculates its traveling cost by using the Dubin's curve based on its location and the task location. The available UAVs are sorted based on their traveling costs and the UAV leader computes small sized coalitions of aerial vehicles to accomplish the task.

Real-world tasks involve inter-task constraints (precedence dependencies), intra-task constraints (task deadlines), and spatial constraints. The previous coalition formation algorithms do not address well the real-world constraints that require complete satisfaction when creating task coalitions. The system robots need to perform planning and scheduling in order to accomplish tasks. A centralized anytime algorithm based on Mixed Integer Linear Programming (MILP) was developed to solve the multi-robot coordination problem and was applied to search and rescue (Koes et al., 2005). This work bridged a major gap in the multi-agent community by finding near optimal solutions for problems involving path planning, online scheduling, and tasks with temporal and spatial constraints. Task environments were discretized into grids and represented as nodes in a weighted graph, where each edge weight represented the robot's traveling cost between the associated nodes. Floyd and Warshalls' shortest path algorithm (Floyd, 1962) was applied to the weighted graph to compute shortest routes to the assigned tasks. The framework generated: (1) high quality task coalitions for midsized robot teams, and (2) optimal task schedules satisfying temporal and spatial constraints, given enough computational time.

Ramchurn et al. (2010) presented the *Coalition Formation with Spatial and Temporal constraints problem* and demonstrated that the problem is *NP*-hard. Ramchurn et al.'s algorithm is similar to Koes et al.'s centralized multi-robot coordination architecture in that it solved the constraint problem using MILP; however, the former approach addresses task deadlines that are not considered in the latter. An anytime algorithm was devised to calculate agent coalitions in real-time.

The challenging task of searching an exponential search space to find the optimal coalition for a given task motivated research to draw cues from nature-inspired biological systems that offer distributed fault-tolerance and adaptive responsiveness towards dynamic environments. This increasing trend of solving complex, optimization problems by employing biologically inspired, metaheuristic algorithms are inspired by social behaviors of insects (e.g., bees, ants, wasps) (Ren et al., 2008; Xia et al., 2004), flocking in birds, school of fish, and social animals (e.g., dolphins) (Haque and Egerstedt, 2009). The next subsection highlights such greedy, swarm-based coalition formation algorithms.

II.1.2.1 Biologically inspired Coalition Formation Algorithms

Particle swarm optimization, a swarm-based technique mimicking behaviors, such as bird-flocking has been used for coalition formation. A collection of potential problem solutions (called *particles*) move through the problem search space, with some domain specific mathematical models governing the *particles*' positions and velocities, to ultimately converge towards the optimal solution. A two-dimensional discrete particle swarm based coalition formation algorithm was presented that derived agent coalitions and permitted overlapping coalitions (Xu and Li, 2008). Zhang et al. (2010) recently presented another particle swarm based coalition formation algorithm that also allows overlapping coalitions, but improves upon its predecessor (Xu and Li, 2008) by proposing a new technique to repair a large number of invalid encodings that appear during the optimization process. Zhang et al.'s algorithm outperformed Xu and Li's approach in relation to solution quality and computational time.

Ant colony optimization techniques are probabilistic approaches based on the social behavior of ants (Dorigo et al., 2006). The first coalition formation algorithm based on ant colony optimization used the inverse of the distance between each agent as a heuristic component (Xia et al., 2004). This heuristic motivates agents that are spatially close to create coalitions. The algorithm was shown to generate coalitions of higher utilities in similar computational time when compared to those computed by a heuristic based coalition algorithm (Shehory and Kraus, 1998). An extended and improved ant colony based coalition algorithm was proposed to generate high quality agent coalitions (Ren et al., 2008). This algorithm attempted to minimize the loss of robot resources and the heuristic is dependent on the task and the communication cost between every pair of connected robots. Compared to the coalitions produced by the previous algorithm (Xia et al., 2004), Ren et al.'s algorithm generated coalitions of higher quality.

Evolutionary algorithms are stochastic, population-based algorithms that replicate biological evolution (e.g., cell reproduction, gene selection) (Ashlock, 2006). Two coalition formation algorithms have been proposed based on cooperative coevolution and the distributed island based approach, respectively (Service, 2009). When benchmarked against RACHNA (Vig and Adams, 2006a), a market-based coalition formation algorithm, the evolutionary algorithms required an order of magnitude fewer communication messages. However, RACHNA outperformed both the proposed distributed evolutionary algorithms in terms of the solution quality (coalition structure utility) and computational time.

Haque and Egerstedt (2009) proposed a decentralized framework for multi-level robot coalitions that emulates the alliance formation process in social animals, such as the bottlenose dolphins. The dynamic interaction of agents (i.e., dolphins) are modeled using hybrid automatons, where agents form dynamic proximity graphs based on nearest neighbor connections. The agent coalitions or alliances are computed based on three coefficient factors: (1) *association*, (2) *rejection*, and (3) *familiarity*. The experimental simulations showed coalition formation among only five agents.

The aforementioned heuristic algorithms sacrifice coalition quality in order to reduce computational time. Another major drawback is that the applicability of heuristics is contingent upon the availability of the required knowledge, which restricts the applicability of heuristic-based algorithms in a wide variety of real-world situations. This shortcoming of any heuristic algorithm inspires the discussion of approximation algorithms that guarantee solution quality.

II.1.3 Approximation Algorithms

This section briefly describes the state-of-the-art approximation algorithms in the multi-agent literature. Despite the close correlation between the coalition formation and the coalition structure generation problems, the former differs in some aspects: (1) specific mission tasks need to be accomplished, and (2) unlike the characteristics function games, the complete set of all possible coalition values is not available. Coarsely, the approximation algorithms can be partitioned into anytime and design-to-time algorithms.

II.1.3.1 Anytime Algorithms

Anytime algorithms are guaranteed to generate quality solutions within some error bounds and permit premature termination. These algorithms generate initial solutions that are within a bound from the optimal solution, but given more time, the solution quality increases progressively as more and more search spaces are examined. However, anytime algorithms suffer from a very high worst case run-time of $O(n^n)$, where n represents the number of agents (Sandholm et al., 1999).

One of the first anytime algorithms for the coalition structure generation problem showed that the problem is NP-complete because $O(2^{n-1})$ coalition structures are examined in order to generate guaranteed quality solutions (Sandholm et al., 1999). The coalition structure space was represented as a graph, with each node representing a particular coalition structure that comprised of multiple agent coalitions. The anytime algorithm searched the two lowest levels of the coalition structure graph and performed a depth first search from the root of the graph to generate solutions.

Another anytime algorithm for coalition structure generation was proposed that produced solutions within a finite bound from the optimal solution (Dang and Jennings, 2004b). The algorithm was similar to Sandholm et al.'s anytime algorithm in that it leveraged the same coalition structure graph. However, Dang and Jennings' anytime algorithm was shown to be extremely fast during the generation of quality solutions with tighter bounds.

A novel, integer-partition based anytime algorithm was proposed by Rahwan et al. (2009b) for coalition structure generation that guaranteed high quality solutions quickly. The algorithm used a new integer-partition based search space representation, where possible coalition structures were categorized based on the sizes of the contained coalitions. Moreover, a branch and bound technique was leveraged to prune large portions of the search space based on the calculated upper and lower value bounds on each coalition structure sub-space, given the coalition values.

II.1.3.2 Design-to-time Approaches

Unlike anytime algorithms, design-to-time algorithms are required to run to completion in order to generate optimal solutions. However, design-to-time algorithms have a better worst case run-time of $O(3^n)$, where n represents the number of agents (Rothkopf et al., 1998). One of the earliest design-to-time algorithms for characteristic function games modeled the computational problem as an evaluation of combinational bids (Rothkopf et al., 1998). The advantage of this algorithm is that, given n agents, the optimal coalition structure is generated in $O(3^n)$ run-time by leveraging the *Dynamic Programming* construct. However, the algorithm suffers from two disadvantages: (1) necessity to run to completion, and (2) significant amount of memory space requirements.

Rahwan and Jennings (2008a) proposed a design-to-time algorithm that addressed the memory issues by leveraging the *Improved Dynamic Programming* construct. Experiments showed that Rahwan and Jennings' algorithm generated quality solutions using fewer operations that required a significantly lower memory space. However, the *Improved Dynamic Programming* algorithm does not permit premature termination and needs to run to completion to compute optimal solutions.

A hybrid approach exploiting the advantages of both the design-to-time and anytime algorithms was proposed (Rahwan and Jennings, 2008b). More recently, a design-to-time, r ($r > 2$) factor approximation algorithm was presented that generates high quality coalition structures with better worst case run-time (Service and Adams, 2011b). Service and Adams' algorithm can also be used as an anytime algorithm by using standard dynamic programming and extracting approximate solutions at appropriate times, thereby giving rise to an anytime algorithm that is capable of generating optimal solutions in $O(3^n)$ time.

Owing to the exorbitant worst case run-time complexities, approximation algorithms are inadequate for real-time critical mission situations. For instance, the *Anytime Integer Partition Algorithm* requires about 2.3 hours to compute optimal coalition structures for just 27 agents, while a *Dynamic Programming*-based approach takes about 2 months for the same computation (Rahwan, 2007).

II.1.4 Market-based Distributed Architectures

Market-based, multi-robot architectures offer effective task allocations in a distributed fashion by leveraging auctions and the most commonly used auctioning approach is based on the Contract Net Protocol (Smith, 1980). The protocol is modeled on the contracting mechanism that is widely used in real-life economics. The agent that discovers a task acts as the *manager/auctioneer* and announces the task details to potential agents (*contractors*) that are connected in the contract net. The *contractors* submit their cost bids for the announced task; the manager evaluates the bids and selects the *contractor* with the lowest bid for the task.

One of the earliest market-based distributed, fault-tolerant architectures was MURDOCH (Gerkey and Matarić, 2002), an auction-based task allocation system. MURDOCH incorporated a resource centric publish/subscribe messaging protocol, where messages are addressed by *subject*, rather than *destination*. The *subject namespace* contained physical robot resources (e.g., camera, gripper, laser), high-level roles (e.g., door-opener, foraging), and abstracted robot states (e.g., busy, idle). Each robot subscribed to a set of subjects that represented their respective capabilities/resources. Task allocations were calculated through auctioning, with the *auctioneer* robot broadcasting the list of required subjects. The system robots (*contractors*) submitted their cost bids for the task and the best bidder is assigned the task. A major limitation was MURDOCH's assumption that real-world tasks can be sub-divided into multiple single-robot, independent tasks.

The Contract Net Protocol (Smith, 1980), when applied to a multi-agent settings does not hold any information (*who* and *when*) regarding the particular agent that can initiate the auctioning process. This issue was addressed in a distributed task allocation system for unmanned aerial vehicles by leveraging a *token-ring* methodology (Lemaire et al., 2004). Uniform distribution of task workload across all the robots was achieved by introducing the *equity coefficient* factor, which ensured the minimization of the traveling time and task workload. A random agent initiated the auctioning process for a task by creating a *token*, thereby becoming the temporary auction leader (*manager*). A *token circulation* process in order to pass the *token* was performed when another agent with a higher *equity coefficient* was available to bid for the same task. Therefore, the task allocation was achieved using standard auctions, but with no static auctioneer. The agent that minimized the traveling cost and task workload was awarded a given task. The system addressed inter-task constraints by employing hierarchical, constraint task trees and the decentralized planning was modeled as a time constrained Multiple Traveling Salesman Problem. Similar to MURDOCH, this system considered only single-robot task.

CoMutaR (**Co**alition formation based on **Mu**lti-**ta**sking **R**obots), a distributed fault-tolerant framework was proposed to handle multi-robot coordination and task allocation (Shiroma and Campos, 2009). The robots' capabilities were expressed as actions, while tasks were expressed as a set of robot actions. The framework allowed virtual sensor sharing by incorporating share-restricted resources, which is very similar to ASyMTRe's schema-based approach (Tang and Parker, 2005). Share-restricted resources can either belong to robots (e.g., processing power, battery energy level, robot's pose) or to task environments (e.g., communication bandwidth). Single-round auctioning was used to compute robot coalitions for tasks. Robots, by virtue of their actions, were capable of generating overlapping coalitions. Once coalitions were formed, coordination among robots was maintained by incorporating *constraint* functions. Absence of a central planner rendered CoMutaR fault-tolerant and scalable, two important requirements of real-world missions with a large number of heterogeneous robots.

DEMiR-CF (**D**istributed and **E**fficient **M**ult**i**-**R**obot-**C**ooperation **F**ramework) was presented as an auction-based, incremental multi-robot framework designed for dynamic task allocation in complex real-world settings that involved inter-related tasks (Sariel-Talay et al., 2011). The distributed framework formulated the Cooperative Mission Achievement Problem and the Coordinated Task Selection Problem, both motivated by models in Operations Research. The key features of DEMiR-CF include: (1) an efficient representation of missions, (2) online task scheduling and planning, (3) task allocation based on auctions, and (4) task reallocation and team reorganization during contingencies. Modeled as finite state machines, each robot possessed information of the world, mission tasks, and every other robot. DEMiR-CF offered fault tolerance by incorporating *Plan B Precaution Routines* to check validation of coalitions continuously. The framework was successfully demonstrated in Navy missions. The major limitation of auction-based architectures is that there is no guarantee on the solution quality.

The first, auction-based distributed task allocation scheme that guaranteed solution quality was Prim Allocation, a simple and fast 2-approximation algorithm for allocating multi-robot tasks (Lagoudakis et al., 2004). Multi-round auctions were held with robots bidding for mission tasks by utilizing their traveling costs as bid amounts. Prim Allocation leveraged Prim's algorithm (Prim, 1957), a greedy approach to compute the minimum spanning tree of a given weighted graph. Each robot modeled the task allocation problem as a Traveling Salesman Problem with the tasks as nodes in a weighted graph. The execution of the Prim Allocation algorithm generated a minimum spanning forest, which was a collection of minimum-cost trees computed by each agent for the given tasks. The scalability and applicability of Prim Allocation to real-world domains with a large number of robots and tasks was justified by its low time complexity, which was formulated as $O((n+m)mlog_2m)$ for n robots and m tasks. However, Prim Allocation was demonstrated to work only with virtual agents in static and dynamic simulated environments.

Crucial to the auctioning procedure is the bidding process. All of the reviewed market-based architectures feature robots as the bidders for a set of tasks. Auction bidders need to have a global view of the available resources in order to improve their individual profits. During complex missions, the global information is often unavailable; therefore, RACHNA (**R**obot **A**llocation through **C**oalitions using **H**eterogeneous **N**on-Cooperative **A**gents) (Vig and Adams, 2006a) reversed the traditional bidding process by having the tasks bid for robots using their utilities. Moreover, RACHNA exploited the resource redundancies in real robots to generate more computationally tractable task allocation solutions. The resource redundancies of robots enabled RACHNA to categorize resources (e.g., laser, camera, bumpers) as services/capabilities (e.g., pushing, foraging, object-tracking). Tasks bid on a set of required services, rather than individual robots. The smaller number of service classes reduced RACHNA's computational time. The task allocation process comprised of two types of software agents: (1) *Service Agents* that served as mediator agents through which tasks bid for services, and (2) *Task Agents* that bid for services on behalf of respective tasks. RACHNA allows task preemption by rerunning the bidding process, but suffers from high communication bandwidth requirements.

Distributed auctioning architectures are not always a panacea because market-based auctioning introduces added complexities involving negotiation protocols, definition of cost functions, and higher communication overhead. TraderBots (Dias, 2004), a novel market-based multi-robot coordination architecture chose a hybrid solution by exploiting the advantages of both centralized and distributed architectures in an opportunistic fashion. The system consisted of a team of self-interested robots aimed at maximizing individual profit. Each robot housed a *Robot Trader* that traded on behalf of the robot for the tasks in the auctioning process and each task had an assigned reward that was achieved on the task's completion. An operator trader (*OpTrader*) played the role of a computational agent and a central software interface between a human operator and the robot team. Under normal circumstances, the robots used their corresponding *Robot Traders* to bid for real-world tasks in a decentralized fashion; thereby, rendering TraderBots highly robust to robot failures and improving scalability. When sufficient time is available, TraderBots appointed either the *OpTrader* or any one particular *Robot Trader* with sufficient processing power as a leader to generate optimal global team plans from individual robots' sub-plans. Therefore, in order to maximize overall utility, TraderBots exploited the advantages of both the centralized and distributed approaches in an opportunistic fashion. Moreover, TraderBots achieved fault tolerance by incorporating *escape clauses/broken deals* in trading during the auctioning process, while allowing task reallocation under contingencies through rebidding processes.

Market-based approaches require high communication bandwidth, and therefore are inappropriate in real-world environments with constrained communications. Given the disadvantages of existing approaches, real-world missions may have a greater propensity towards using greedy algorithms that can generate solutions of acceptable quality within the stipulated time. This dissertation develops two greedy, swarm-based search algorithms that are applicable to *NP*-complete optimization problems, including the multi-agent coalition formation for real-world applications, but without using common heuristics, such as constraining the coalition sizes.

II.2 Ant Colony Optimization Algorithms

ACO algorithms have been investigated thoroughly for several *NP*-complete problems, such as Traveling Salesman Problem (TSP) (Dorigo and Gambardella, 1997; Dorigo et al., 1996; Stutzle and Hoos, 2000), resource-constrained scheduling (Merkle et al., 2002), classification (Martens et al., 2007), multi-robot coalition formation (Ren et al., 2008; Xia et al., 2004), and data mining (Xie and Mei, 2007). Forager ants wander around the nest searching for food and upon finding good food, they retrieve the food and travel back to their nest, while depositing chemicals, *pheromones*. Additional forager ants retrace the path to the food source by sensing the pheromones, and over time more and more ants converge on the shortest path from the nest to the food source. Ant colony based algorithms utilize this foraging behavior to generate shortest path solutions in optimization problems satisfying various constraints.

The effectiveness of most ACO algorithms has been demonstrated via applying them to the TSP, which is one of the most widely studied combinatorial optimization problems. Given n cities modeled as a weighted, undirected graph, $G(V,E,W)$, with the cities represented as the graph nodes, the objective is to identify a closed tour of the shortest length that visits each city once and only once. Each edge, $E_{ij} \in E$ connecting two cities, $V_i, V_j \in V$ is associated with a non-negative weight, $w_{ij} \in W$ that represents the distance between the two cities. A symmetric TSP is a special variant of the problem, where $\forall i, j\ w_{ij} = w_{ji}$.

Ant System, the first ACO algorithm developed, simulated a set of artificial ants with limited memory and computational capabilities that emulate the foraging behavior in ant swarms (Dorigo et al., 1996). Each ant, ϕ starts at a random city or node in the graph, $G(V,E,W)$ and incrementally generates a city tour. The ϕ^{th} ant located at city i utilizes the transitional probability, P_{ij}^{ϕ} to probabilistically select the next city, j to be added to its current tour. The transitional probability is defined as

$$P_{ij}^{\phi} = \begin{cases} \frac{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{\Sigma_{j \in \mathscr{N}^i} \tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}} & \text{if } j \in \mathscr{N}^i \\ 0 & \text{otherwise,} \end{cases} \tag{II.3}$$

where τ_{ij} and η_{ij} represent the pheromone level on the edge $E_{ij} \in E$ and the heuristic information that motivates the transition, respectively. The parameters α and β regulate the impact of the pheromone and heuristic values during the transition process. $\mathscr{N}^i$ denotes the unvisited neighborhood of city i. After the end of each iteration, the pheromone levels on all the graph edges are lowered by using the pheromone evaporation rate, $0 < \rho < 1$ that emulates the natural pheromone evaporation in the real-world due to sunlight. All the ants deposit a small quantity of pheromone (Δ_{ij}^{ϕ}) on their respective tour edges, in accordance with the ant's solution quality (tour length). The pheromone update process at time $t+1$ is defined as

$$\tau_{ij}(t+1) \leftarrow \rho * \tau_{ij}(t) + \sum_{\phi=1}^{m} \Delta_{ij}^{\phi}, \tag{II.4}$$

and

$$\Delta_{ij}^{\phi} = \begin{cases} \frac{Q}{L_\phi} & \text{if ant tour has edge, } E_{ij} \\ 0 & \text{otherwise,} \end{cases} \tag{II.5}$$

where m is the number of ants, Q is a constant and L_ϕ is the length of the tour generated by the ϕ^{th} ant. This algorithm was investigated for a TSP instance with 30 cities.

The *Elitist Ant System* extended the *Ant System* and leveraged an *elite* set of additional ants for depositing pheromones that replicated the current global best solution (Dorigo et al., 1996). The *Elitist Ant System* performed better than the *tabu search* and *simulated annealing* approaches, when applied to a TSP instance comprising 30 cities. The number of elite ants was demonstrated to be equal to the number of cities in the tested problem instance.

The *Rank-based Ant System* incorporated the *Elitist Ant System*'s elitist strategy and a technique that ranks the ants based on their solution quality after every iteration (Bullnheimer et al., 1997). The pheromone update policy uses the current global best solution along with the top ω solutions in the ranking that are weighted according to the respective ant's rank. The *Rank-based Ant System* was compared with the predecessors on five TSP instances with city counts ranging from 30 to 132. The *Rank-based Ant System* performed notably better than the *Ant System*, but comparable to the *Elitist Ant System*.

The Ant Colony System (*ACS*) was the first to leverage only the single best solution for pheromone depositing (Dorigo and Gambardella, 1997). *ACS* differs from its predecessors in a number of ways. First, a pseudo-random-proportional transition rule is leveraged when the ϕ^{th} ant on node i selects the next node, S^* by applying

$$S^* = \begin{cases} \arg\max_{j \in \mathcal{N}^i} \{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}\} & \text{if } q \leq q_0 \\ s & \text{otherwise,} \end{cases} \tag{II.6}$$

where $0 \leq q_0 \leq 1$ is a algorithm parameter, q is a random number uniformly distributed in the range [0,1], and $s \in \mathcal{N}^i$ is probabilistically selected according to Equation II.3. When $q \leq q_0$, *ACS* exploits knowledge from the previous iterations, otherwise the algorithm exhibits an exploration behavior. Second, only the current global-best solution is used to deposit pheromones. Furthermore, a local pheromone update procedure is integrated into the incremental search that encourages search exploration. *ACS* experimentally generated better solutions, when compared to three heuristics algorithms (e.g., simulated annealing, elastic nets (Durbin and Willshaw, 1987), self-organizing maps) on three TSP instances with city counts ranging from 50 to 100.

Pheromone updates using a single best ant solution (either the current global-best or iteration-best) often result in rapid intensification of the associated edge pheromones in the search graph. The exponential pheromone evaporation further constrains the search process. This rapid change in the pheromone levels often results in search stagnation, in which case the algorithm becomes stuck in local optima. The Max-Min Ant System (*MMAS*) bounds the pheromone levels on each graph edge within set limits in order to address search stagnation (Stutzle and Hoos, 2000). *MMAS* features three major components: (1) pheromone depositing by leveraging the single best ant (either the best-so-far or the iteration best), (2) bounded pheromone trails within $[\tau_{min}, \tau_{max}]$, and (3) initializing the algorithm by setting the initial edge pheromones to τ_{max} to promote search exploration. Theoretical proofs illustrate that $\tau_{max} = \frac{1}{(1-\rho)\times f(s^{gb})}$, where $f(s^{gb})$ is the current best solution and ρ is the evaporation rate. *MMAS* outperformed the *ACS*, *Rank-based Ant System*, *Elitist Ant System*, and the original *Ant System* on TSP instances with up to 198 cities.

Motivated by evolutionary algorithms, the *Best-Worst Ant System* (Cordon et al., 2000) incorporates a dual update policy, where graph edges belonging to the current global-best solution receive positive reinforcements, while those belonging to the iteration-worst solution, but not in the global best, are negatively reinforced. The *Population-based Ant Colony Optimization* algorithm (Oliveira et al., 2011) leverages a fixed sized population of solutions, similar to *Genetic Algorithms* (Grefenstette et al., 1985). The graph edges belonging to an ant's solution that enters this population are positively reinforced; however, the components of the solution exiting the population receive negative updates. The pheromone modifications are only performed during the insertion or removal of solutions from the population; therefore, this algorithm has a $O(n)$ pheromone update step, instead of the conventional $O(n^2)$. Despite the reduced computational time and comparable performance to *MMAS*, the performance of the *Population-based Ant Colony Optimization* approach is dependent on the size and update strategy of its population archive and the pheromone re-initialization intervals. Such constraints restrict the algorithm's applicability to a broad spectrum of combinatorial optimization problems.

II.2.1 Decentralized Ant Colony Optimization Algorithms

Contemporary ant colony optimization algorithms are primarily centralized, and thus suffer from longer convergence time and brittleness. Randall and Lewis (2002) presented a master-slave configuration to parallelize the ACO approach, where a cluster of ant(s) acts as a slave and assigned a separate processor. After each iteration, each cluster broadcasts its local solution to a master processor to ensure a global pheromone update. The updated pheromone structure is communicated back to each slave to resume processing. The naive, distributed approach was successfully applied to eight TSP instances, but the achieved speed improvement was neutralized by the high communication overhead.

A more sophisticated, distributed architecture for the optimization technique was presented as the ACODA (Ant Colony Optimization on a Distributed Architecture) framework and applied to the TSP (Ilie and Bădică, 2013). ACODA models the cities as software agents (node) and partitions clusters of the connected cities across multiple computing servers. Ants are modeled as software objects that can be exchanged asynchronously across the software nodes. Each node keeps information about every other city node(s) that are directly connected to it in the traveling map. Preliminary results corroborated the scalability of ACODA, where the computational time reduced with an increase in the number of processing servers.

Another promising method to decentralize ACO is to leverage a set of cooperating multiple ant colonies, where each colony independently attempts to solve the same problem search space, and communicate crucial information among themselves in order to concentrate on the promising search space. Middendorf et al. (2000) proposed and investigated four communication policies that are imperative to the execution time and convergence of this approach. The communication strategies were applied to an instance of TSP with 101 cities. Results show that better quality solutions can be achieved through sharing of the local best solution among colonies that are connected in an unidirectional ring topology. An extensive empirical evaluation for multiple ant colony parallelism with statistical analysis was conducted to investigate the role of various factors, such as number of colonies, communication strategy and topology, and information sharing schedule (Twomey et al., 2010).

II.3 Multi-agent Coordination Architectures

Coordination plays a very crucial role in a team of agents in order to accomplish a given task and coordination architectures provide such solutions for the heterogeneous agents through information exchanges via sensor sharing and behavior coordination. One of the earliest fault-tolerant, distributed, and behavior-based multi-robot coordination architectures was ALLIANCE (Parker, 1998). The software architecture leveraged mathematically modeled motivations, such as *impatience* and *acquiescence* to motivate the robots to perform tasks during contingencies. The motivations enabled the robots to achieve adaptive behavior selection; thereby, rendering the framework responsive to unexpected changes in the environments and robot failures. *Impatience* and *acquiescence* are synonymous to exogenous (a robot detecting faults in neighboring robots) and endogenous (a robot detecting a fault in itself) fault detection techniques, respectively (Christensen et al., 2009). Each robot in the system was pre-programmed with a set of behaviors and the behavior activation was triggered by the motivation models. The architecture was successfully implemented on a robot team performing a hazardous waste cleanup task.

ASyMTRe (**A**utomated **Sy**nthesis of **M**ulti-robot **T**ask solutions through software **Re**configuration) offered sensor sharing across networked robots by utilizing *schemas* (Tang and Parker, 2005). Schemas represent fundamental robot behaviors that are pre-programmed into robots at design time. The architecture used four categories of schemas: (1) environmental, (2) perceptual, (3) motor, and (4) communication. Tasks were defined as a set of required motor schemas along with task specific parameters (e.g., goal position, pushing direction). ASyMTRe leveraged an anytime algorithm to automatically compute necessary connections between sensors and schemas across multiple robots based on *information invariant theory*. A few drawbacks of ASyMTRe include its centralized implementation and its failure to address the sensor constraints introduced when sharing sensor capabilities during task execution.

Zhang and Parker (2010) proposed the IQ-ASyMTRe (Information Quality based ASyMTRe), a distributed framework for tightly-coupled tasks that extended ASyMTRe to address the latter's introduction of sensor constraints during the sensor sharing process by incorporating information quality to model the sensor constraints.

Chaimowicz et al. (2003) presented ROCI (**R**emote **O**bjects **C**ontrol **I**nterface), a distributed programming framework designed for multi-robot perception and control. Each robot was considered to be a computational node containing several ROCI modules (e.g., processing, sensing). The modules can be considered as data processing blocks encapsulating specific functions. A ROCI kernel running on every robot had three responsibilities: (1) acquiring updated information from the rest of the robots and the environment, (2) providing inter-agent communication, and (3) facilitating dynamic connections between several ROCI modules across the robot nodes. The ROCI modules were connected using the ROCI pin architecture, a network transparent wiring construct that provided abstracted communication endpoints for the modules based on data types (similar to ASyMTRe). Complex tasks were expressed as instances of collections of ROCI modules and accomplished by connecting the inputs and outputs of the required modules across the available robots. The effectiveness of the ROCI architecture was successfully demonstrated by deploying a team of air and ground robots in urban environments (Chaimowicz et al., 2005).

A highly adaptable architecture, TeamCore was presented for heterogeneous human-robot teams (Tambe et al., 2000). Each agent (robot, agent, or person) in the team was represented by TeamCore proxy, which comprised STEAM (a **S**hell for **TEAM**work) (Tambe, 1997), a generic teamwork model for automating coordination among agent proxies. The TeamCore proxies enabled agents to adapt to dynamic teams comprised of multiple heterogeneous agents. Team adaptation was provided by four methods: (1) adaptive autonomy, (2) adaptive execution, (3) adaptive monitoring, and (4) adaptive information delivery. Human developers interacted with the agents using the Team-Oriented programming interface to provide a hierarchical mission decomposition. The interface facilitated the interaction with KARMA (Knowledgable Agent Resources Manager Assistant), an agent resource manager that managed agents and their resources. Given the Team-Oriented program, the agent proxies collectively derived plans to accomplish the mission tasks.

Freedy et al. (2008) proposed the multi-agent framework, MAAF (Multiagent Adjustable Autonomy Framework) for human-robot teams to accomplish complex, tactical missions that leveraged the TeamCore architecture. MAAF exploits diverse capabilities of heterogeneous team agents that comprises Robots, Agents, and Persons (RAP) and uses an organization hierarchy to capture the team agents depending on their authority. Moreover, MAAF proposes to employ standard artificial intelligence models, such a Markov Decision Process to address uncertainties in real-world missions. However, MAAF has been proposed as a concept, with simulation results showing humans in supervisory roles.

II.4 Multi-robot System Taxonomy

A major contribution of this dissertation is *i-CiFHaR*, which selects the most appropriate coalition formation algorithm(s) to apply to a given mission by leveraging *influence diagrams* that compare the algorithms across multiple mission criteria. A multi-robot system taxonomy is necessary that defines multiple dimensions/features along which algorithms can be classified and compared. This section describes some of the taxonomies designed for multi-robot systems.

Gerkey and Matarić proposed one of the earliest and most widely cited taxonomies that was designed for task allocation in multi-robot systems (Gerkey and Matarić, 2004). However, the taxonomy dimensions are highly abstracted and not broad enough to classify coalition algorithms effectively. The three taxonomy dimensions are:

1. single-task (ST) versus multi-tasks (MT) robots, depending on whether a robot can perform a single task or multiple tasks simultaneously;
2. single-robot (SR) versus multi-robots (MR) tasks, based on whether a task requires a single robot or multiple robots to get accomplished; and
3. instantaneous assignment (IA) versus time-extended (TE) assignments of tasks, with the time-extended allocations requiring long term planning and scheduling for future tasks.

A more effective multi-robot systems' taxonomy was presented that classifies situations based on communication and computational capabilities (Dudek et al., 1996). The taxonomy dimensions are:

1. collective size, denoting the number of agents in the system;
2. communication range, denoting the maximum communication range between two agents;
3. communication topology, the connectivity network of the agents;
4. communication bandwidth, representing the maximum number of messages permitted;

5. collective reconfigurability, the rate at which coalition agents can reconfigure themselves during contingencies;

6. processing ability, the computation model of each robot; and

7. team composition, specifying whether coalition agents are homogeneous or heterogeneous.

A taxonomy approach providing theoretical bases for cooperative multiple robots was presented (Cao et al., 1997). The taxonomy dimensions include:

1. group architecture, capturing the characteristics of the system agents (e.g., homogeneous, heterogeneous, aware, unaware) and the agents' communication structure;

2. resolve conflicts, enabling multiple robots to coordinate and cooperate during task execution, while sharing a common workspace;

3. cooperation origin, the agent cooperation structure (e.g., social animals, game theory);

4. learning, rendering systems adaptive to environment dynamics; and

5. geometric aspect (e.g., path planning, group formations).

A task oriented multi-robot taxonomy was proposed that was motivated by three task dependent factors (Lau and Zhang, 2003). The dimensions are: (1) task demands, (2) resource constraints, and (3) profit objective. When the agent resources are constrained or limited, then all the tasks cannot be accomplished; therefore, the best sub-set of tasks needs to be selected that maximizes system utility.

A hierarchical taxonomy was proposed that comprised of two major dimensions (coordination and system) at the top level (Farinelli et al., 2004). The coordination dimension, which captures the degree of agent cooperation is further divided into three classes:

1. knowledge, capturing agent awareness;

2. coordination, required in tightly coupled tasks; and

3. organization level, denoting the centralized or distributed robot architecture.

The system dimension is divided into four classes:

1. communication topology;

2. team composition, denoting agent types;

3. system architecture, representing system's level of responsiveness towards environmental changes; and
4. team size.

While these taxonomies exist, they do not provide a comprehensive taxonomy capturing a board set of axes/dimensions for the multi-robot task allocation problem. Service and Adams (2010) presented such a taxonomy with dimensions that were partitioned into four major relation-based groups: (1) agent, (2) task, (3) domain, and (4) algorithm. Additionally, Service and Adams classified a number of coalition formation algorithms according to the taxonomy dimensions. This broad set of dimensions motivated the use of Service and Adams's taxonomy for classifying the coalition formation algorithms for the research presented in this dissertation. The taxonomy dimensions are:

1. agent orientation, describing whether an agent is group-rational or self-interested;
2. agent heterogeneity, denoting the types of agents (e.g., homogeneous, heterogeneous);
3. agent capability, representing whether the capabilities of an agent are described in terms of resources (e.g., sonars, lasers, camera) or services/roles (e.g., foraging, watching, box-pushing);
4. agent awareness, the ability of an agent to understand other team agents and their activities;
5. agent structure, the agents' connectivity network;
6. inter-task constraint, describing whether tasks have inter-dependencies, such as precedence ordering;
7. task preemption, denoting whether an algorithm allows the preemption of an ongoing task upon the arrival of a high priority task;
8. task requirement, describing the requirements based on a resource/service model;
9. intra-task constraint, denoting that tasks have duration times and completion deadlines;
10. task coupling, the degree of coordination required by the team to accomplish a given task;
11. performance criterion, the objective function that needs to be optimized to find quality task coalition solutions;
12. communication requirements, denoting the bounds on the number of inter-agent messages;
13. task allocation, describing whether an instantaneous or time-extended task assignment is required;
14. spatial constraints between real-world robots and tasks;

15. overlapping coalition, requirement of overlapping teams to reduce resource losses;

16. algorithm technique, describing whether the coalition formation algorithm employs a greedy, approximation, or an auction-based approach to calculate task coalitions;

17. algorithm implementation, denoting whether the system agents calculate task coalitions in a distributed fashion or a single, central agent computed the coalitions (centralized technique); and

18. algorithm constraints, such as constraints on coalition sizes.

CHAPTER III

The Hybrid Ant Colony Optimization Algorithms

This chapter introduces two hybrid swarm based search algorithms, called the *sA-ANT* and *sA-ANT** (***s**imulated **A**nnealing*-inspired ***ANT*** colony optimization) that leverage novel pheromone update policies, which deviate significantly from the conventional techniques in order to address the search stagnation drawback of existing state-of-the-art ACO algorithms. Search stagnation in ACO approaches lead to sub-optimal solutions. The pheromone update methodology is an important aspect in any ACO approach, because it incorporates positive reinforcements of the promising regions in the search space. Such positive reinforcements lead to incremental learning that guides the search process towards better solutions, as an ACO algorithm progresses.

This dissertation develops two pheromone depositing policies by integrating the simulated annealing technique that result in enhanced search exploration during the initial phase of the algorithm, followed by improved exploitation during the later phases. The effectiveness of the developed ant algorithms has been demonstrated by successful application to three combinatorial optimization problems, the Traveling Salesman Problem, maximal clique problem, and the multi-agent coalition formation problem. A distributed variant of *sA-ANT** was also developed and applied for the multi-robot coalition formation problem.

III.1 The *sA-ANT* algorithm

sA-ANT employs *Dynamic Searching*, where a dynamically modulating number of ants explore a larger search area during the initial algorithm iterations and gradually converge towards good solutions over time; thereby, effectively resulting in a balanced search exploration and exploitation. The pheromone update policies of the state-of-the-art ACO algorithms (Dorigo and Gambardella, 1997; Ren et al., 2008; Stutzle and Hoos, 2000) leverage either the iteration-best or the current global-best solution for updating the pheromones. However, positive reinforcements by the single best solution, along with the exponential pheromone evaporation quickly results in *search stagnation*, i.e., the search space surrounding these current best solutions are only exploited in later iterations, which results in a sub-optimal solution. *sA-ANT* incorporates the *simulated annealing* technique to dynamically modulate the number of ants that deposit pheromones, which increases exploration during the initial iterations, and exploitation during the later phases. Moreover, this dynamic pheromone deposit scheme increases the likelihood of generating higher quality solutions, without stagnating in local optima.

Simulated annealing (Kirkpatrick et al., 1983) is an iterative, stochastic search method for optimization problems and is motivated by annealing in metallurgy, where metals are heated and cooled in a controlled fashion in an attempt to increase their strength. The simulated annealing algorithm begins with an initial random solution, S_{t-1} in the search space, followed by the random selection of a new solution, S_t in the neighborhood of the previous solution during every iteration. The new random solution is accepted with a probability $Pr(S_{t-1}, S_t, Temp)$[1], which is defined as,

$$Pr(S_{t-1}, S_t, Temp) = \begin{cases} 1 & f(S_t) > f(S_{t-1}) \\ exp^{-\Delta E/Temp} & \text{otherwise,} \end{cases} \tag{III.1}$$

where $\Delta E = f(S_{t-1}) - f(S_t)$ is the energy difference in the solution qualities; $f(x) : x \to \mathbb{R}$ is the function that provides the solution quality; and $Temp$ is the annealing temperature. The annealing temperature is initialized to a high value and is lowered in a controlled fashion using the *annealing schedule* after each iteration. The high annealing temperature allows lower quality solutions to be accepted stochastically initially; however, with the gradual lowering of the temperature, more poor solutions are discarded. The high initial temperature allows the search process to avoid local optima and to widely explore the search space. Despite allowing a wide exploration of the search space, the simulated annealing technique's randomness suffers from longer computational time for solution convergence.

sA-ANT is presented in Algorithm 1. The algorithm starts by initializing the pheromone levels of every edge in a search graph, $G(V, E, W)$ to an initial value, τ_0. During each iteration of the algorithm, a set of ants generates a collection of *nAnts* solutions. Each ant, ϕ starts with a random graph node, $random\_node \in V$ and incrementally generates its solution, S_ϕ by stochastically adding graph nodes (*next_node*) to S_ϕ, until the ant solution satisfies the given problem. During each incremental addition, the ϕ^{th} ant selects the most appropriate graph node, *next_node* to add to its partial coalition S_ϕ in accordance with the transitional probability, which is defined as

$$P^{\phi}_{current,j} = \begin{cases} \frac{\tau^{\alpha}_{current,j} \times \eta^{\beta}_{current,j}}{\sum_{j \in \mathscr{N}^{current}} \tau^{\alpha}_{current,j} \times \eta^{\beta}_{current,j}} & \text{if } j \in \mathscr{N}^{current} \\ 0 & \text{otherwise,} \end{cases} \tag{III.2}$$

where α, β are the ACO parameters that control the impact of the pheromone (τ) and heuristic (η) information, respectively; and $\mathscr{N}^{current}$ represents the neighboring nodes of the current graph node, *current*. The original *sA-ANT* algorithm (Sen and Adams, 2013a), developed for the multi-robot coalition formation problem, leveraged a very aggressive transitional probability, similar to the *Ant Colony System* (Dorigo and Gambardella, 1997) that was defined as:

$$R^{*}_{j} = \underset{R_j \in \mathscr{N}^{current} \setminus S_\phi}{\text{argmax}} (\tau^{\alpha}_{current,j} \times \eta^{\beta}_{current,j}), \tag{III.3}$$

[1]$Pr(S_{t-1}, S_t, Temp)$ is defined for a maximization problem

where R_j^* is the next node in the graph that is selected in a greedy fashion; α, β are the ACO parameters; $\tau_{current,j}$ and $\eta_{current,j}$ denote the pheromone and heuristic values, respectively for the current graph node, *current*. The disadvantage of this approach is that the search becomes restricted very quickly and results in poor search exploration. Therefore, the current *sA-ANT* algorithm (Algorithm 1) leverages Equation III.2 in order to enhance the search exploration, as used in contemporary ACO approaches, in addition to the integration of the simulated annealing technique.

At the end of each iteration, all the ant solutions generated during the present iteration are evaluated for pheromone updates. Each ant, ϕ calculates the acceptance probability (Pr_ϕ) of its solution, S_ϕ, which is calculated for a maximization problem as,

$$Pr_\phi = \begin{cases} 1 & f(S_\phi) > f(S_{global\_best}) \\ exp^{\Delta E_\phi / anneal\_temperature} & \text{otherwise,} \end{cases} \tag{III.4}$$

where $\Delta E_\phi = f(S_\phi) - f(S_{global\_best})$, S_{global_best} is the current global best solution, and $anneal\_temperature$ is the annealing temperature. Similarly, Pr_ϕ for a minimization problem is derived as,

$$Pr_\phi = \begin{cases} 1 & f(S_\phi) < f(S_{global\_best}) \\ exp^{\Delta E_\phi / anneal\_temperature} & \text{otherwise,} \end{cases} \tag{III.5}$$

where $\Delta E_\phi = f(S_{global\_best}) - f(S_\phi)$, and $anneal\_temperature$ is the annealing temperature. Based on the acceptance probability, Pr_ϕ, the ϕ^{th} ant updates or positively reinforces the pheromones of all the members of its solution, S_ϕ.

When an ant generates a better solution than the current global best solution, then the ant always updates the pheromone levels of its solution's members with probability 1; thereby, ensuring that the members of the most current global best solution are reinforced. Otherwise, if the quality of an ant's solution is lower than the current global best solution, then this solution is accepted with an acceptance probability value, Pr_ϕ less than 1, which is proportional to the energy difference, ΔE_ϕ and the annealing temperature, $anneal\_temperature$. During the initial iterations, the annealing temperature is high; therefore, almost all ant solutions are accepted owing to their high probabilities. This high annealing temperature facilitates rapid exploration of a large search space, without stagnating in a local optima. The annealing temperature is lowered in a controlled fashion with every iteration resulting, over time in only the acceptance of high quality solutions and enhanced exploitation. The ϕ^{th} ant updates the pheromones of all its solution members according to:

Algorithm 1 The *sA-ANT* algorithm

Input: α; β; Evaporation rate, ρ; Number of iterations, *nIter*; Number of ants, *nAnts*; Initial annealing temperature, $Temp_0$; Annealing schedule rate, γ; AM_G- Adjacency Matrix of graph, $G(V,E,W)$; Number of nodes in graph, n

Output: Anytime solution, S_{global_best}

```
1: Initialize pheromone on each graph edge, $E_{ij} \in E$ for edge based transition strategy OR each graph node, $V_i \in V$ for vertex based transition strategy
2: $iteration\_solution\_list \leftarrow \emptyset$    // Stores all ant solutions during an iteration
3: $S_{iteration\_best} \leftarrow \emptyset$    // Iteration-best solution
4: $S_{global\_best} \leftarrow \emptyset$    // Global-best solution
5: $S_\phi \leftarrow \emptyset$    // $\phi^{th}$-ant solution
6: $number\_of\_ants\_depositing \leftarrow 0$    // Number of ants depositing pheromones
7: $anneal\_temperature \leftarrow Temp_0$    // Simulated annealing temperature
8: for iterations = 1 to $nIter$ do
9:    for $\phi$ = 1 to $nAnts$ do
10:       $random\_node \leftarrow rand(1,n)$
11:       // Each ant incrementally creates a solution, $S_\phi$
12:       $S_\phi \leftarrow random\_node$
13:       $current \leftarrow random\_node$
14:       while (!IsSolutionComplete($S_\phi$)) do
15:          $next\_node \leftarrow V_j' \in V \setminus S_\phi$
16:          $current \leftarrow next\_node$
17:          $S_\phi \leftarrow S_\phi \cup next\_node$
18:       end while
19:       Update $S_{iteration\_best}$    // Modify the iteration best solution
20:       $iteration\_solution\_list \leftarrow iteration\_solution\_list \cup \{(\phi, S_\phi)\}$
21:       $S_\phi \leftarrow \emptyset$
22:    end for
23:    Update $S_{global\_best}$    // Modify the global best solution
24:    Evaporate pheromones of all graph edges
25:    for $i$ = 1 to $iteration\_solution\_list.size()$ do
26:       $Pr_\phi \leftarrow$ CalculateAcceptanceProbability($iteration\_solution\_list[i]$)
27:       $accept_\phi \leftarrow Pr_\phi > rand(0,1)$
28:       if $accept_\phi$ then
29:          Update pheromone trails using this ant solution
30:          $number\_of\_ants\_depositing \leftarrow number\_of\_ants\_depositing + 1$
31:       end if
32:    end for
33:    if $number\_of\_ants\_depositing == 0$ then
34:       Update pheromone trails using $S_{global\_best}$
35:    end if
36:    Pheromone Bounding within $\tau_{max}$ and $\tau_{min}$
37:    $iteration\_solution\_list \leftarrow \emptyset$
38:    $anneal\_temperature \leftarrow anneal\_temperature \times \gamma$    // Lowering of annealing temperature
39:    $number\_of\_ants\_depositing \leftarrow 0$
40: end for
41: Return $S_{global\_best}$
```

$$\tau_{ij} = \begin{cases} \tau_{ij} + Pr_{\phi} \times \psi & (i,j) \in S_{\phi} \\ 0 & \text{otherwise,} \end{cases} \tag{III.6}$$

where τ_{ij} is the pheromone level of the graph edge and ψ is the amount of pheromone deposited by ϕ^{th} ant.

For instance, $\psi = \frac{1}{L_{\phi}}$ for the TSP, where L_{ϕ} is the length of the tour generated by the ϕ^{th} ant. Since, the acceptance of an ant's solution is probabilistic, the amount of pheromone deposited is proportional to the acceptance probability. During the initial algorithm phases, a poor solution, even when accepted stochastically is allowed to deposit a large amount of pheromones on its members. However, during the later phases, as the annealing temperature is reduced, despite a stochastic selection of such a poor solution, the amount of pheromone deposited will be curtailed substantially based on the solution's acceptance probability.

An additional pheromone deposit is performed occasionally on the members of S_{global_best}, which ensures that the respective members of the current global best solution are reinforced, even when no ant solution is stochastically selected during a particular iteration. Finally, the pheromone levels of the graph edges are bounded within an upper and lower threshold, τ_{Max} and τ_{Min}, as defined for the *Max-Min Ant System* (Stutzle and Hoos, 2000). The bounding of pheromones avoids both excessive depositing and the evaporation of the pheromones.

III.2 The *sA-ANT** algorithm

The *sA-ANT** algorithm incorporates a completely different pheromone update policy than *sA-ANT*'s update policy. *sA-ANT** draws cues from *evolutionary algorithms*, specifically *genetic algorithms* that emulate the natural evolution through selection, crossover, and mutation (Grefenstette et al., 1985). *Genetic algorithms* leverage a fixed sized population of candidate solutions, each consisting of specified properties. During each iteration, the candidate solutions are evaluated based on quality, and the superior solutions are probabilistically selected to generate new child candidate solutions through crossover and mutation. The pool of candidate solutions are iteratively refreshed to accommodate better solutions over time. However, a major shortcoming of *genetic algorithms* lies in the population size, which heavily affects their performance and requires careful tailoring depending on the problem domains.

*sA-ANT** maintains a repository of good solutions whose size is dynamically modulated during the search process. Each ant solution has an associated fitness value that is computed using the simulated annealing mechanism. Unlike *sA-ANT*, where a dynamically modulating number of ants are permitted to deposit pheromones, *sA-ANT** probabilistically selects the single best fit ant solution to deposit pheromone based on its solution quality. Furthermore, the algorithm permits quality based stochastic forgetting of poor solutions in the repository. Equipped with a high initial annealing temperature and a slow annealing schedule, the algorithm avoids stagnating in a local optima; thereby, addressing the undesirable search stagnation problem commonly encountered with existing ACO approaches.

The *sA-ANT** algorithm is presented in Algorithm 2. At the end of each iteration, the current iteration-best solution is inserted into the repository and is characterized by a fitness or quality metric, which is defined as the *acceptance probability* and is computed using the *Simulated Annealing* technique (see Equations III.4 and III.5). The iteration-best solution is only inserted into the repository, provided it is not in the current repository (i.e., only unique solutions are maintained in the repository).

At the end of each iteration, *sA-ANT** performs two procedures. First, a forgetting process is performed, where bad solutions are eliminated stochastically from the repository based on their quality (*acceptance probabilities*) with respect to the current global best solution. Second, the *acceptance probabilities* for the remaining solutions are updated based on the current global-best solution, and the simulated annealing temperature and schedule. The forgetting of poor solutions serves two crucial objectives: (1) it prevents probabilistic selection of poor solutions for updating pheromones over time, and (2) prohibits the continuous increase in the repository size. Thus, *sA-ANT**'s repository is dynamically modulated. Finally, only the single best fit ant solution from the repository is selected stochastically for depositing pheromones based on each solution's *acceptance probability*. *sA-ANT** begins with a high annealing temperature; therefore, during the initial algorithm phase, all the ant solutions have very high and mostly equivalent fitness qualities. This mechanism allows the flexibility of using even a poor solution during the initial iterations, which avoids constraining the search based on the current global or iteration best solution. Due to the gradual annealing schedule, the search exploration is enhanced by prohibiting the concentration of pheromone levels of the graph edges belonging to the single best ant solution. As the annealing temperature gradually decreases, over time only good solutions in the repository will have higher *acceptance probabilities*; thereby, increasing their likelihood of being stochastically selected for pheromone deposits.

III.3 Modeling Appropriate Annealing Parameters

The performance of *sA-ANT** and *sA-ANT*, like any other existing ACO algorithm is dependent on multiple ACO parameters. A number of existing ACO algorithms have identified good ranges for these ACO parameters (e.g., α, β, ρ); therefore, this dissertation leveraged these existing parameter values. However, the performance of both *sA-ANT* and *sA-ANT** depends on their integrated simulated annealing mechanism, thus models for calculating good approximate values of the annealing parameters (e.g., initial annealing temperature, annealing schedule, γ) are presented in order to ensure that both hybrid algorithms can be applied to a wide range of optimization problems.

Algorithm 2 The *sA-ANT** algorithm

```
Input: $\alpha$; $\beta$; Evaporation rate, $\rho$; Number of iterations, $nIter$; Number of ants, $nAnts$; Initial annealing
    temperature, $Temp_0$; Annealing schedule rate, $\gamma$; $AM_G$- Adjacency Matrix of graph, $G(V,E,W)$; Number
    of nodes in graph, $n$
Output: Anytime solution, $S_{global\_best}$
 1: Initialize pheromone on each graph edge, $E_{ij} \in E$ for edge based transition strategy OR each graph node,
    $V_i \in V$ for vertex based transition strategy
 2: $Repository_0 \leftarrow \emptyset$                    // Dynamically modulated repository of solutions
 3: $S_{iteration\_best} \leftarrow \emptyset$                    // Iteration-best solution
 4: $S_{global\_best} \leftarrow \emptyset$                    // Global-best solution
 5: $S_\phi \leftarrow \emptyset$                    // $\phi^{th}$-ant solution
 6: $anneal\_temperature \leftarrow Temp_0$                    // Simulated annealing temperature
 7: for iterations = 1 to $nIter$ do
 8:   for $\phi$ = 1 to $nAnts$ do
 9:     $random\_node \leftarrow rand(1,n)$
10:     // Each ant incrementally creates a solution, $S_\phi$
11:     $S_\phi \leftarrow random\_node$
12:     $current \leftarrow random\_node$
13:     while (!IsSolutionComplete($S_\phi$)) do
14:       $next\_node \leftarrow V_j' \in V \setminus S_\phi$
15:       $current \leftarrow next\_node$
16:       $S_\phi \leftarrow S_\phi \cup next\_node$
17:     end while
18:     Update $S_{iteration\_best}$                    // Modify the iteration best solution
19:     $S_\phi \leftarrow \emptyset$
20:   end for
21:   Update $S_{global\_best}$                    // Modify the global best solution
22:   Evaporate pheromones of all graph edges
23:   $Pr_{iteration\_best} \leftarrow$ CalculateAcceptanceProbability($S_{iteration\_best}$)
24:   // Check for duplicate solutions in repository
25:   if !IsSolutionPresentInRepository($S_{iteration\_best}$) then
26:     $Repository_{iterations} \leftarrow Repository_{iterations-1} \cup \{(S_{iteration\_best}, Pr_{iteration\_best})\}$
27:   end if
28:   //Stochastic Forgetting of bad solutions from $Repository$
29:   for k = 1 to $Repository_{iterations}.size()$ do
30:     $Repository_{iterations}[k].Probability \leftarrow$ UpdateAcceptProbability($Repository_{iterations}[k].Solution$)
31:     if $Repository_{iterations}[k].Probability < rand(0,1)$ then
32:       Delete $Repository_{iterations}[k]$
33:     end if
34:   end for
35:   Select $S^+ \in Repository_{iterations}$                    // Stochastic solution selection for pheromone update
36:   Update pheromone trails using $S^+$
37:   Pheromone Bounding within $\tau_{Max}$ and $\tau_{Min}$
38:   $anneal\_temperature \leftarrow anneal\_temperature \times \gamma$          // Lowering of annealing temperature
39: end for
40: Return $S_{global\_best}$
```

Definition 3. *Let the initial annealing temperature be $Temp_0$, and S_{global_best} and S_{approx} be the best known global solution and an approximate solution, respectively for a given optimization problem. The acceptance probability of an approximate solution with respect to the best known problem solution is given by Equations III.4 and III.5. An ideal initial value of $Temp_0$ permits the acceptance probability of S_{approx} with respect to S_{global_best} to be equal to an admission ratio, θ. Therefore, $Temp_0 = \frac{S_{global\_best} - S_{approx}}{ln\theta}$ for a minimization problem, and $Temp_0 = \frac{S_{approx} - S_{global\_best}}{ln\theta}$ for a maximization problem.*

Lemma 1. *Let the number of algorithm iterations be $nIter$, the simulated annealing schedule rate be γ, and the starting annealing temperature be $Temp_0$. Then $\gamma = exp^{ln(Temp_0^{-1})/nIter}$.*

Proof. The annealing schedule rate is derived by constraining the annealing temperature to be reduced to 1 at algorithm completion. The gradual lowering of the annealing temperature is defined by $Temp_0 \times \gamma^{nIter} = 1$, which leads to $\gamma = exp^{ln(Temp_0^{-1})/nIter}$. □

Example 1. Consider a TSP instance *eil*51 (Reinelt, 2014) that defines a TSP of 51 cities. Let $S_{approx} = 600$ be an approximate tour length solution for the problem, as computed by the nearest neighbor algorithm. The best known solution for this instance is $S_{global\_best} = 426$ (obtained from the ***TSPLIB*** library (Reinelt, 2014)). Assume an *admission ratio*, $\theta = 0.95$. Therefore, from Definition 1, the initial annealing temperature is calculated as, $Temp_0 = \frac{426-600}{ln0.95} = 3392.2$. Once the initial temperature is calculated, its corresponding annealing schedule is derived using Lemma 1 as, $\gamma = exp^{ln(3392.2^{-1})/5000} = 0.998$, assuming the number of algorithm iterations is 5000.

III.4 Experimental Design and Results

The performance of *sA-ANT** and *sA-ANT* is demonstrated for three classes of optimization problems that include a minimization problem, the *Traveling Salesman Problem*, and two maximization problems: *maximal clique* and *multi-agent coalition formation*.

III.4.1 Experimental Design for the Traveling Salesman Problem

The performance of the *sA-ANT** and *sA-ANT* algorithms for the TSP was compared to three state-of-the-art ACO algorithms that leverage a single ant depositing policy, similar to *sA-ANT**. The three algorithms are: *Max-Min Ant System* (*MMAS*) (Stutzle and Hoos, 2000), *Ant Colony System* (*ACS*) (Dorigo and Gambardella, 1997), and *Best-Worst Ant System* (*BWAS*)[2] (Cordon et al., 2000). The standard ACO software package, ***ACOTSP.V1.03*** (ACO-Software, 2014) provides implementations for several ACO algorithms, including the *MMAS*, *ACS*, and *BWAS*. However, the *MMAS* and *ACS* algorithms were also implemented separately in C++

[2]BWAS leverages two ants to deposit pheromones, the iteration best and the iteration worst ants

using the Qt framework on an Ubuntu machine. ***ACOTSP.V1.03***'s implementations of the *MMAS*, *ACS*, and the *BWAS* use several integrated methods, such as local search (e.g., 2-OPT, 3-OPT) techniques to address any search stagnation. Since, *sA-ANT** and *sA-ANT* seek to maximize the search exploration capability of ACO approaches without leveraging any such techniques, the local searches were disabled for all three ACO algorithms. The Qt-implementation of the *MMAS* and *ACS* did not incorporate any local search mechanisms for the said reasons.

The ***TSPLIB*** library (Reinelt, 2014) provides several symmetric TSP instances with various city counts in XML format along with their optimal solutions. A TSP instance from ***TSPLIB*** has the format "*nameXXX*", where *name* is the instance name, while XXX indicates the number of cities in the instance. Five TSP instances (*bays29*, *eil51*, *eil76*, *kroA100*, and *d198*) were used for the experiments, with the number of cities ranging from 29 to 198. *sA-ANT** and *sA-ANT* were compared to both the Qt-implementation and ***ACOTSP.V1.03***'s implementation of the *MMAS* and *ACS*; however, only ***ACOTSP.V1.03***'s implementation of *BWAS* was used for the experiments.

A number of parameters required careful tuning for the best performance of each algorithm. The parameters of each of the three ACO algorithms (for both the Qt-implementation and ***ACOTSP.V1.03***'s implementation) were set to values as defined in their respective original implementations. The parameters α and β were set to 1.0 and 2.0 for each of the three algorithms in accordance with their original implementations. The remaining parameters for the *MMAS* were set as defined in its original implementation (Stutzle and Hoos, 2000), with $m = n$, where m is the number of ants and n is the number of cities in the problem instance, evaporation rate, ρ= 0.98, $\rho_{best} = 0.05$ for the calculation of τ_{min}, and $\tau_0 = \frac{1}{(1-\rho)\times L_{nn}}$, where L_{nn} is the tour length generated by the nearest-neighbor heuristic. The *ACS* algorithm parameters were set according to the original publication (Dorigo and Gambardella, 1997), with $\rho = 0.9$, $q_0 = 0.9$, $m = 10$, and $\tau_0 = (n \times L_{nn})^{-1}$. The parameter settings for the *BWAS* were $\rho = 0.8$, $q_0 = 0.8$, $m = 25$, and $\tau_0 = \frac{n}{L_{nn}}$, based on its original implementation (Cordon et al., 2000). The *MMAS* algorithm is currently the most efficient ACO algorithm; therefore, the parameters for *sA-ANT** and *sA-ANT* were chosen to match those of *MMAS*, with $\alpha = 1.0$, $\beta = 2.0$, $\rho = 0.98$, $m = n$, $\rho_{best} = 0.05$, and $\tau_0 = \frac{1}{(1-\rho)\times L_{nn}}$. The same number of tours were constructed for each of the five algorithms, which was fixed to $10000 \times n$, where n is the number of cities in the instance and a candidate list of size 20 was leveraged for all the algorithms, as defined in the original *MMAS* implementation (Stutzle and Hoos, 2000). Moreover, *sA-ANT** and *sA-ANT* require annealing parameters. The initial annealing temperature ($Temp_0$) and annealing schedule (γ) during every trial were calculated based on Definition 1 and Lemma 1 with admission ratio, $\theta = 0.95$ and $S_{approx} = L_{nn}$. Since ACO algorithms are stochastic in nature, all algorithms were run 25 times and the means and standard deviations were averaged over all trials. Two hypotheses are analyzed for the TSP experiments.

- H_1 : The mean tour lengths of *sA-ANT** and *sA-ANT* will be significantly lower than the remaining three ACO algorithms.
- H_2 : *sA-ANT** and *sA-ANT* will exhibit higher search exploration capabilities than *MMAS*.

III.4.1.1 Experimental Results

The hybrid algorithms and the three ACO algorithms for the TSP experiments were compared based on (1) Generated tour lengths, and (2) Search exploration capability.

III.4.1.1.1 Tour Lengths

The mean tour lengths and standard deviations for each ACO algorithm across the five TSP instances over 25 trial runs are shown in Table III.1. Both *sA-ANT** and *sA-ANT* outperformed all other algorithms consistently over all instances by providing the lowest mean tour lengths and the minimum standard deviation. *MMAS* performed third best. The results in Table III.1 demonstrate that for the four problem instances with city counts ranging from 29 to 100, *sA-ANT** successfully generated tour lengths within 0.15% of the optimal solution, and within 0.86% of the optimal tour for the largest instance with 198 cities. These percentage deviations are calculated by subtracting the optimal tour lengths for each TSP instance from the mean tour lengths provided by *sA-ANT** and dividing the result with respective optimal tour lengths that are obtained from the ***TSPLIB*** library (Reinelt, 2014).

The tour lengths generated by the Qt-implementations of *MMAS* and *ACS* were either better or on par with those computed by $ACOTSP.V1.03$'s *MMAS* and *ACS*; therefore, the statistical tests were conducted using the results from the Qt-implementation of *MMAS* and *ACS*. Shapiro-Wilk Normality Test was conducted to determine the normality distribution of the results over the 25 trials. Since the distribution was not normal, a Kruskal-Wallis one-way analysis of variance determined that there is a significant main effect of algorithm on the generated tour lengths ($\chi^2 = 14.86, df = 3, p < 0.001$). Individual comparisons were performed using the Wilcoxon rank-sum test, and the results are provided in Table III.2. *sA-ANT** generated significantly shorter tour lengths for all five TSP instances, while *sA-ANT* generated significantly shorter tour lengths for all the problem instances, but *eil*51. Compared to *sA-ANT*, *sA-ANT** generated shorter tour lengths for all the instances, except *bays*29, where both computed the optimal tours. The standard deviations of the tour lengths generated by *sA-ANT** were the smallest when compared to remaining algorithms.

Problem			Qt-Implementation		ACOTSP.V1.03		
Instances	sA-ANT*	sA-ANT	MMAS	ACS	MMAS	ACS	BWAS
bays29	**2020**	**2020**	2021.9	2022	2021.9	2022.1	2023.1
Opt: 2020	**(0)**	**(0)**	(3.46)	(3.98)	(3.5)	(3.8)	(5.1)
eil51	**426.7**	427.3	427.4	428.3	427.6	428.1	429.4
Opt: 426	**(0.85)**	(0.98)	(1.42)	(2.93)	(1.5)	(2.5)	(3.6)
eil76	**538.3**	538.72	539.3	541.6	539.6	541.5	543.3
Opt: 538	**(0.98)**	(1.6)	(1.6)	(3.6)	(2.2)	(3.9)	(3.8)
kroA100	**21295.5**	21296.52	21330.2	21405.3	21330.3	21379.3	21405.4
Opt: 21282	**(28.0)**	(32.1)	(49.2)	(145.5)	(43.2)	(160.3)	(152.6)
d198	**15919.3**	15937.48	15984.6	15988.9	15993.8	16127.9	16276.5
Opt: 15780	**(44.3)**	(59.3)	(55.9)	(82.1)	(58.8)	(113.9)	(226.9)

Table III.1: Mean tour lengths for the Traveling Salesman Problem instances. The standard deviation is presented in parenthesis. The best solutions are highlighted in bold.

Problem Instances	**sA-ANT*** vs **MMAS**	**sA-ANT** vs **MMAS**	**sA-ANT*** vs **ACS**	**sA-ANT** vs **ACS**	**sA-ANT*** vs **BWAS**	**sA-ANT** vs **BWAS**
bays29	$W = 400.0$, $z = -2.8$, $\mathbf{p < 0.01}$	$W = 400.0$, $z = -2.8$, $\mathbf{p < 0.01}$	$W = 387.5$, $z = -2.5$, $\mathbf{p < 0.01}$	$W = 387.5$, $z = -2.5$, $\mathbf{p < 0.01}$	$W = 458.5$, $z = -2.9$, $\mathbf{p < 0.01}$	$W = 458.5$, $z = -2.9$, $\mathbf{p < 0.01}$
eil51	$W = 419$, $z = -2.2$, $\mathbf{p = 0.014}$	$W = 300$, $z = -0.2$, $p = 0.39$	$W = 432$, $z = -2.4$, $\mathbf{p < 0.01}$	$W = 306.5$, $z = -1.3$, $p = 0.4$	$W = 458$, $z = -2.9$, $\mathbf{p < 0.01}$	$W = 380.5$, $z = -1.3$, $\mathbf{p < 0.01}$
eil76	$W = 480$, $z = -3.7$, $\mathbf{p < 0.01}$	$W = 442$, $z = -2.8$, $\mathbf{p < 0.01}$	$W = 533$, $z = -4.5$, $\mathbf{p < 0.01}$	$W = 491$, $z = -3.73$, $\mathbf{p < 0.01}$	$W = 570$, $z = -5.3$, $\mathbf{p < 0.01}$	$W = 550$, $z = -4.9$, $\mathbf{p < 0.01}$
kroA100	$W = 432$, $z = -2.5$, $\mathbf{p < 0.01}$	$W = 428$, $z = -2.5$, $\mathbf{p < 0.01}$	$W = 538.5$, $z = -4.5$, $\mathbf{p < 0.01}$	$W = 540.3$, $z = -4.6$, $\mathbf{p < 0.01}$	$W = 501$, $z = -3.8$, $\mathbf{p < 0.01}$	$W = 499$, $z = -3.8$, $\mathbf{p < 0.01}$
d198	$W = 512$, $z = -3.86$, $\mathbf{p < 0.01}$	$W = 459.5$, $z = -2.9$, $\mathbf{p < 0.01}$	$W = 475.5$, $z = -3.1$, $\mathbf{p < 0.01}$	$W = 428.5$, $z = -2.2$, $\mathbf{p < 0.01}$	$W = 622$, $z = -6.0$, $\mathbf{p < 0.01}$	$W = 618$, $z = -5.93$, $\mathbf{p < 0.01}$

Table III.2: Wilcoxon Rank Sum test results for comparing tour lengths generated by the algorithms. $p < 0.05$ is significant, while $p < 0.01$ is highly significant.

III.4.1.1.2 Search Exploration Capability

The efficiency of each ACO algorithm is determined by the effectiveness of its search capability. The pheromone update policies of *sA-ANT** and *sA-ANT* incorporate simulated annealing and seek to enhance the search exploration capability of the algorithms. A key metric that highlights the search exploration capability is $\lambda-$branching factor ($b\_factor(r)$) of a node, r (Gambardella et al., 1995), which is defined as

$$b\_factor(r)_\lambda = \sum_{s=1}^{|\mathcal{N}^r|} \tau_{rs} > (min(\tau_r) + \lambda \times (max(\tau_r) - min(\tau_r))), \tag{III.7}$$

where $\lambda \in \{0.04, 0.06, 0.08\}$ is a constant, τ_{rs} is the pheromone amount of the edge that connects nodes r and $s \in \mathscr{N}^r$, $\mathscr{N}^r$ represents the neighbors of node r, while $min(\tau_r)$ and $max(\tau_r)$ denote the minimum and maximum pheromone values of all the edges connected to node r. The average branching factor is calculated as $\frac{\sum_{r=1}^{n} b\_factor(r)_\lambda}{2\times n}$, where n is the number of cities in the graph. The extensiveness of the search exploration of *sA-ANT** and *sA-ANT* was compared only to the *MMAS* algorithm, because *MMAS* performed best among the previously existing ACO algorithms, and the same pheromone bounding mechanism is leveraged to address search stagnation, thereby providing a fair comparison.

sA-ANT and *sA-ANT** provide superior search exploration capability over *MMAS* on TSP instances, *eil51* (Fig. III.1), *eil76* (Fig. III.2), *kroA100* (Fig. III.3), and *d198* (Fig. III.4). *MMAS* enters search stagnation, which is identified by the average branching factor of 1.00 after ≈ 300 iterations for the *eil51*, *eil76*, and *kroA100* instances. Similarly, *MMAS* enters stagnation at ≈ 500 iterations for the *d198* instance. *sA-ANT** continues to search, even beyond 1000 iterations.

Compared to *sA-ANT**, the average branching factor of *sA-ANT* remains very high for a longer duration of the algorithm. Unlike *sA-ANT**, which leverages a stochastically selected single best fit ant to deposit pheromones in every iteration, *sA-ANT* dynamically modulates the number of ants that deposit pheromones. During the initial phase of the algorithm, a larger number of ants are used to deposit pheromone that facilitates higher search exploration; however, with the gradual decrease in the annealing temperature, the number of ants responsible for depositing pheromones is reduced, thereby shifting the focus to search exploitation. Figure III.5 demonstrates that the number of ants depositing pheromones for three TSP instances are dynamically modulated during the progress of the algorithm, which prevents *sA-ANT* from stagnating in local optima during the initial search phase.

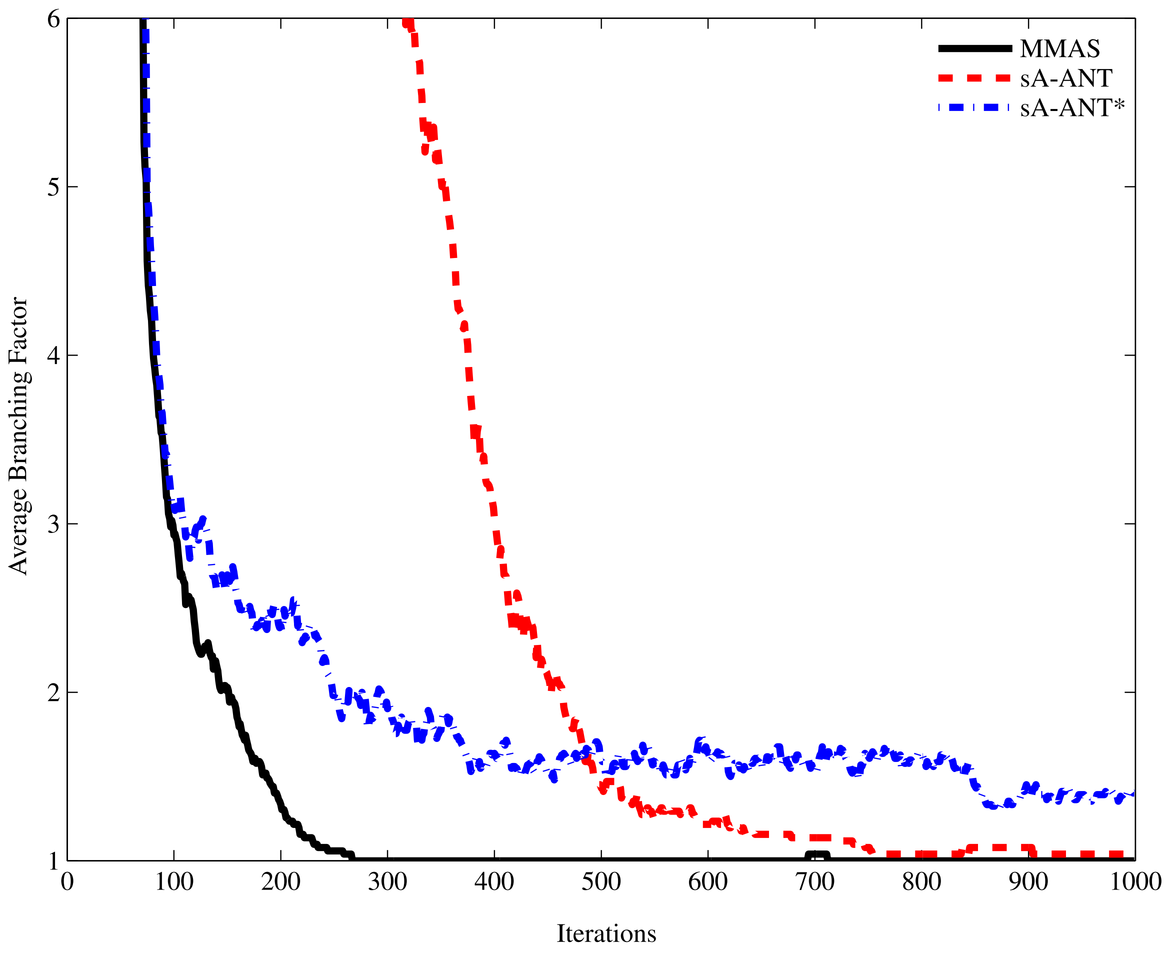


Figure III.1: *sA-ANT*, *sA-ANT**, and *MMAS* are compared based on the Average branching factor vs. iterations for TSP instance, *eil51* ($\lambda = 0.04$).

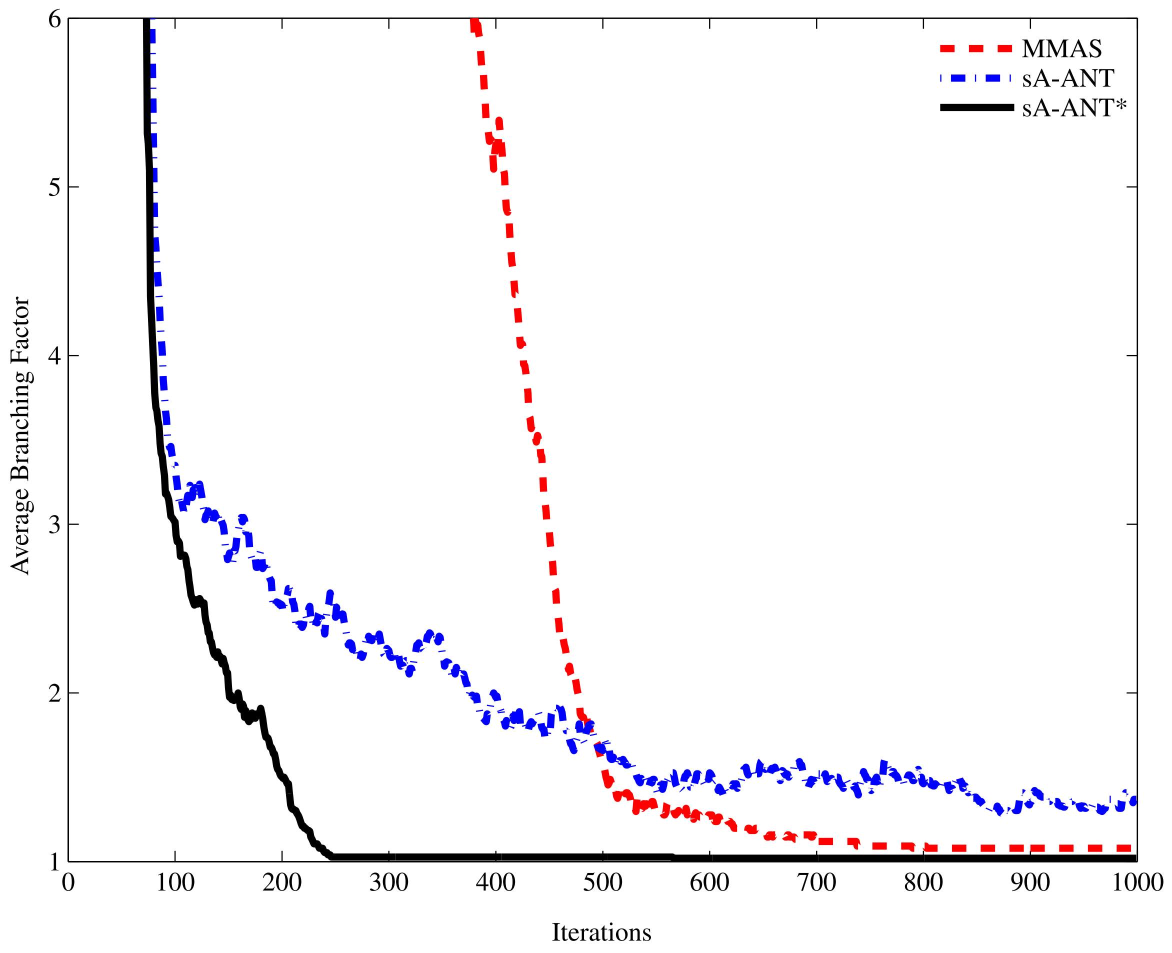


Figure III.2: *sA-ANT*, *sA-ANT**, and *MMAS* are compared based on the Average branching factor vs. iterations for TSP instance, *eil76* ($\lambda = 0.04$).

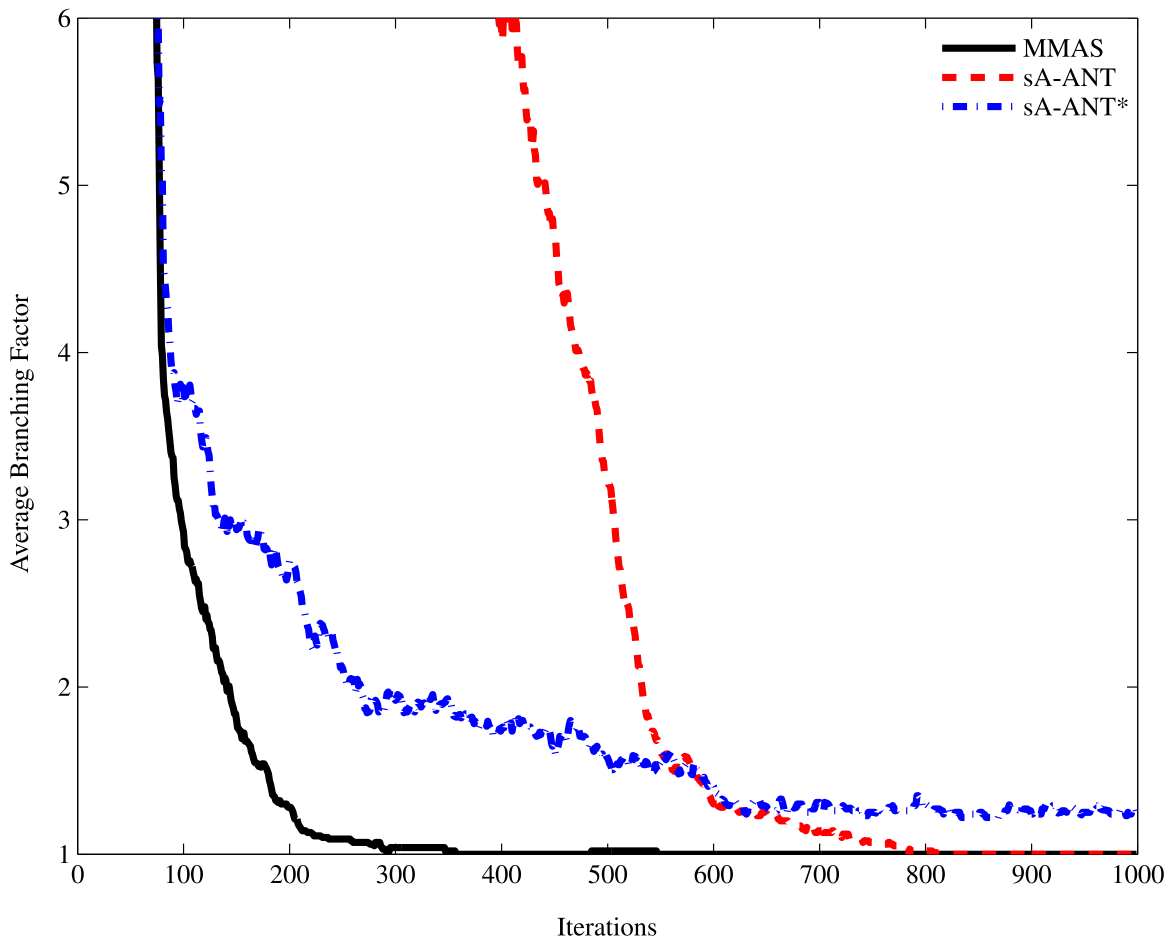


Figure III.3: *sA-ANT*, *sA-ANT**, and *MMAS* are compared based on the Average branching factor vs. iterations for TSP instance, *kroA100* ($\lambda = 0.04$).

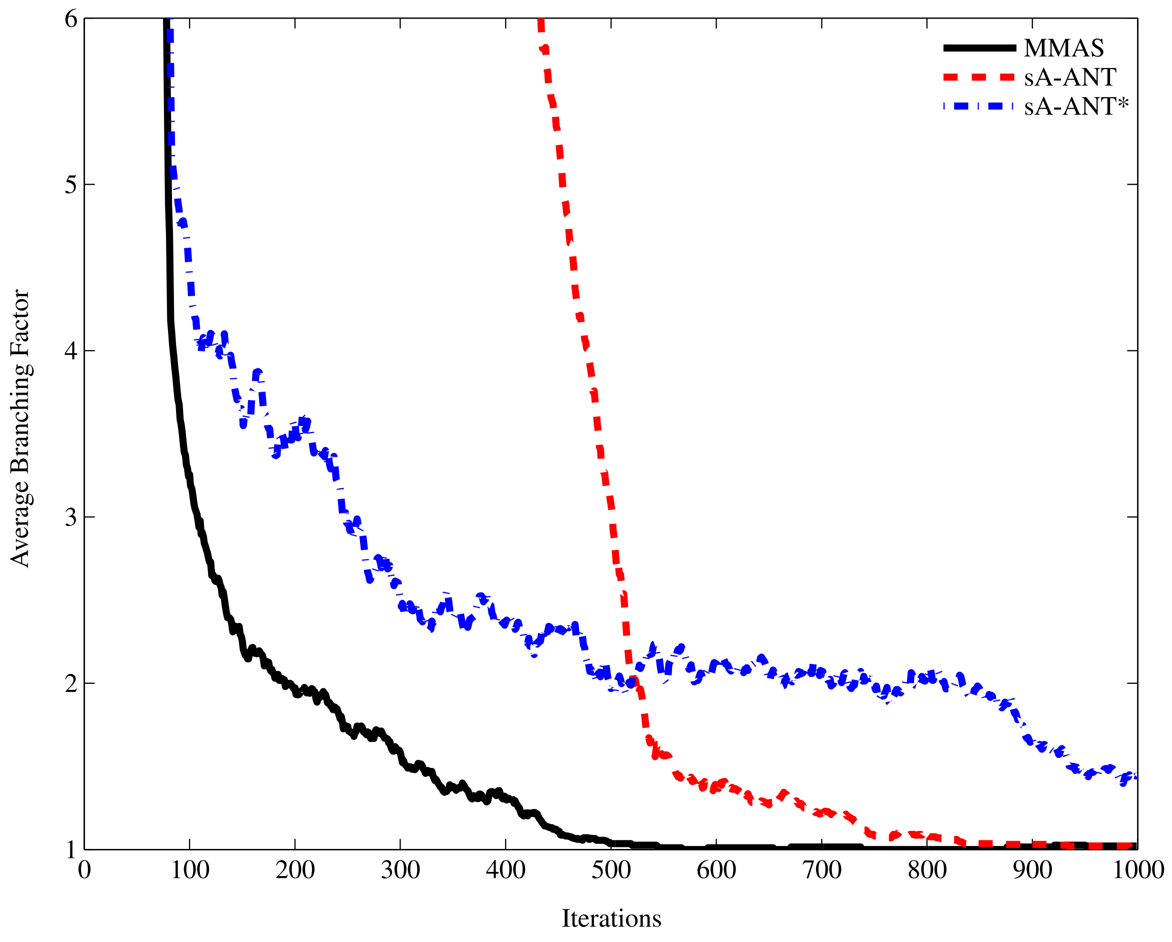


Figure III.4: *sA-ANT*, *sA-ANT**, and *MMAS* are compared based on the Average branching factor vs. iterations for TSP instance, *d198* ($\lambda = 0.04$).

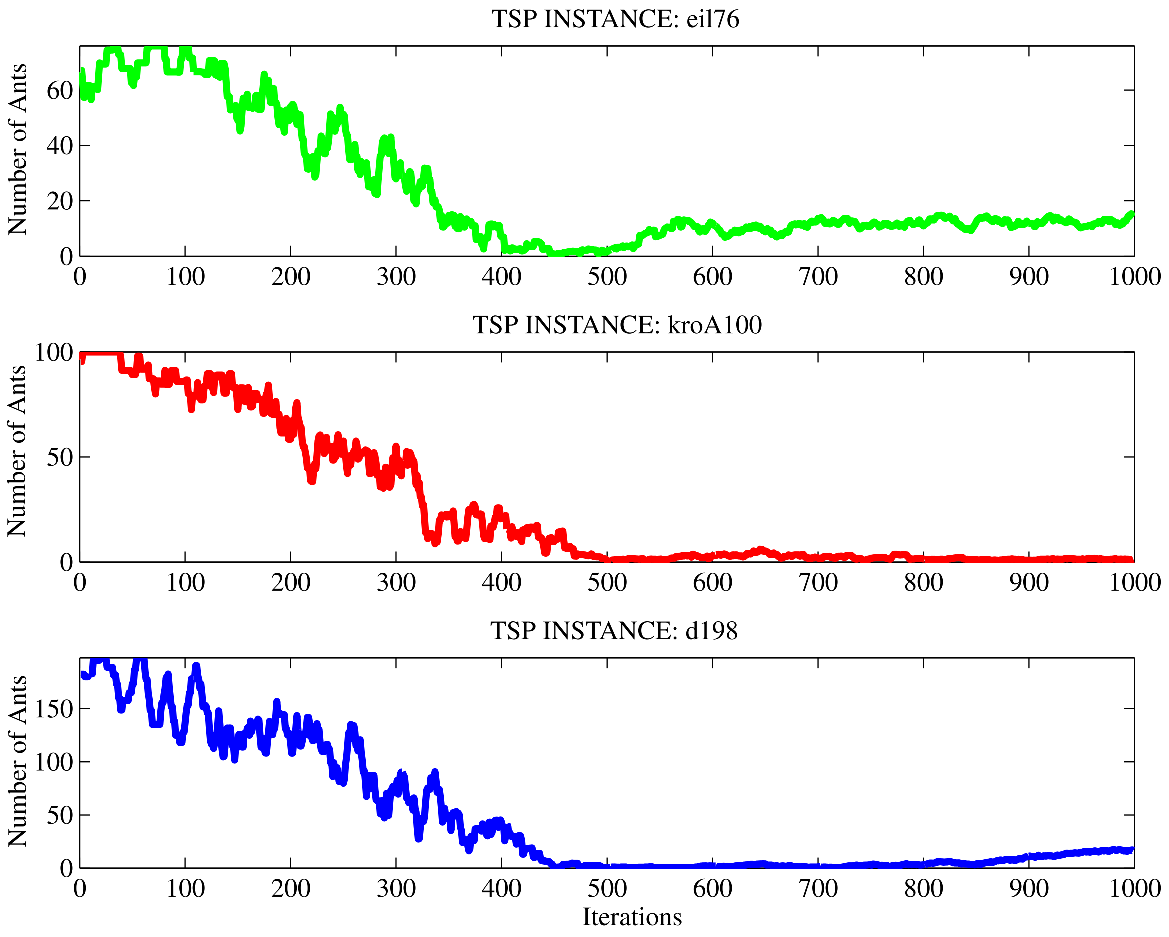


Figure III.5: Illustration of the dynamically modulated number of ants depositing pheromones for *sA-ANT* for three TSP instances.

III.4.1.2 Discussion of TSP results

*sA-ANT** generated significantly shorter tour lengths for all the problem instances with city counts ranging from 29 to 198, while *sA-ANT* computed significantly shorter tour solutions for four TSP instances; thereby, supporting H_1. The results of *sA-ANT** and *sA-ANT* show that both algorithms effectively address the search stagnation drawback of existing ACO algorithms; thereby, supporting hypothesis, H_2. Both the hybrid algorithms maintained a high exploration quotient for a higher number of iterations, when compared to *MMAS*, the best known prior ACO approach.

III.4.2 Experimental Design for Multi-agent Coalition Formation problem

The *sA-ANT** and *sA-ANT* algorithms were compared with two ant-based coalition formation algorithms (Ren et al., 2008; Xia et al., 2004). The *Ant-Coalition* algorithm (Ren et al., 2008) improved upon an existing ACO-based coalition formation approach (Xia et al., 2004), which is henceforth termed as *Ant-Coalition-Basic*. The number of robots was set to $n = 50, 100, 150, 200$ for each mission scenario. Ten mission scenarios were randomly generated, each comprised of a single task requiring five resource types. The resource requirements remained fixed for a particular mission across all the trials. The first mission scenario began with each resource type value set to 15. Each scenario increased the resource type values by 5, to a maximum of 60 for each task requirement value during the final mission. The ACO-based algorithms are non-deterministic, thus the experimental results were averaged over all trials of the same mission.

Twenty five trials of each mission scenario were conducted for each algorithm at each value of *n*. Each trial for a particular mission and *n* value used the same task requirements. The robot locations, task locations and robot capabilities were randomly generated for a particular mission trial and *n* value, but remained the same across all the algorithms. The task and robot locations were generated to lie within a 5000m x 5000m area. The task reward or utility was assigned to $U(T_j) = 5000 \times \sum_{i=1}^{5} TR_{ji}$ in order to maintain uniformity across all missions, trials, and algorithms (T_j is the j^{th} task and TR_{ji} is the i^{th} resource requirement of j^{th} task). The utility of a coalition, S_c is defined as $V(S_c) = U(T_j) - RE(S_c) - CC(S_c) - TC(S_c)$, where $U(T_j)$ is the task reward, $RE(S_c)$ is the resource expenditure of the coalition, $CC(S_c)$ is the communication cost for the coalition members, and $TC(S_c)$ is the total travel distance for the coalition.

Multiple parameters require careful setting in order to achieve better performance. The first set of parameters (α, β, ρ, ε) that regulate the performance of ACO algorithms were set to α=1, β=2, ρ=0.99, ε=0.005, $nAnts$=50, and $nIter$=2000 for *sA-ANT*, *sA-ANT**, and *Ant-Coalition*, as defined in the original implementation of the *Ant-Coalition* algorithm (Ren et al., 2008). The parameter settings for *Ant-Coalition-Basic* included α=1, β=2, ρ=0.90, and $nAnts$=10, along with two additional parameters for its *inner hormone* factor that were set to $g_{max} = 10$ and $N_{circles} = 30$ according to its original implementation (Xia et al., 2004). The values for these parameters were consistent with the prior experiments (Ren et al., 2008; Xia et al., 2004) in order to ensure a valid comparison with the prior results.

Four hypotheses are considered for the TSP experiments.

- H_3 : *sA-ANT** and *sA-ANT* will exhibit a significantly higher search exploration capabilities.
- H_4 : *sA-ANT** and *sA-ANT* will generate significantly higher quality coalitions owing to their greater search exploration capability, compared to *Ant-Coalition* and *Ant-Coalition-Basic*.
- H_5 : *sA-ANT** and *sA-ANT* will generate coalitions that with significantly lower traveling cost for the coalition members, compared to *Ant-Coalition* and *Ant-Coalition-Basic*.
- H_6 : *sA-ANT** and *sA-ANT* will require higher computational time, compared to *Ant-Coalition* and *Ant-Coalition-Basic*.

III.4.2.1 Experimental Results

Four primary metrics were leveraged to compare the results: (1) Unique Coalition Counts, (2) Coalition quality, (3) Traveling distance, and (4) Computational time.

III.4.2.1.1 Number of Unique Coalitions

The number of unique coalitions generated by each algorithm is an important metric that implicitly indicates the extensiveness of an algorithm's search capability. Figure III.6 shows the mean number of unique coalitions generated by each algorithm as the number of robots increases. Both *sA-ANT* and *sA-ANT** generated a significantly higher number of unique coalitions than the *Ant-Coalition* and the *Ant-Coalition-Basic* algorithms, as n increased. A Kruskal-Wallis one-way analysis of variance determined that there is a significant main effect of algorithm on the number of unique coalitions generated ($\chi^2 = 50.95$, $df = 3$, $p < 0.001$). Individual comparisons were performed using the Wilcoxon rank-sum test. When compared to the *Ant-Coalition* and *Ant-Coalition-Basic*, the number of coalitions generated by both *sA-ANT* and *sA-ANT** was significantly higher for $n = 50$, 100, 150, 200 (Table III.3); thereby, indicating that the presented variants of *sA-ANT* exhibit more efficient exploration.

Number of Robots	**sA-ANT*** vs **Ant-Coalition**	**sA-ANT** vs **Ant-Coalition**	**sA-ANT*** vs **AntCoalition-Basic**	**sA-ANT** vs **AntCoalition-Basic**
50	$W = 0$, $z = -4.6$, $\mathbf{p < 0.001}$	$W = 0$, $z = -5.7$, $\mathbf{p < 0.001}$	$W = 0$, $z = -5.7$, $\mathbf{p < 0.001}$	$W = 0$, $z = -5.8$, $\mathbf{p < 0.001}$
100	$W = 0$, $z = -4.6$, $\mathbf{p < 0.001}$	$W = 0$, $z = -5.8$, $\mathbf{p < 0.001}$	$W = 0$, $z = -5.7$, $\mathbf{p < 0.001}$	$W = 0$, $z = -5.8$, $\mathbf{p < 0.001}$
150	$W = 14$, $z = -4.6$, $\mathbf{p < 0.001}$	$W = 0$, $z = -5.6$, $\mathbf{p < 0.001}$	$W = 0$, $z = -5.7$, $\mathbf{p < 0.001}$	$W = 0$, $z = -5.7$, $\mathbf{p < 0.001}$
200	$W = 61$, $z = -2.5$, $\mathbf{p = 0.012}$	$W = 0$, $z = -5.6$, $\mathbf{p < 0.001}$	$W = 0$, $z = -5.6$, $\mathbf{p < 0.001}$	$W = 0$, $z = -5.6$, $\mathbf{p < 0.001}$

Table III.3: Wilcoxon Rank Sum test results for comparing the unique number of coalitions generated by the algorithms. $p < 0.05$ is significant, while $p < 0.01$ is highly significant.

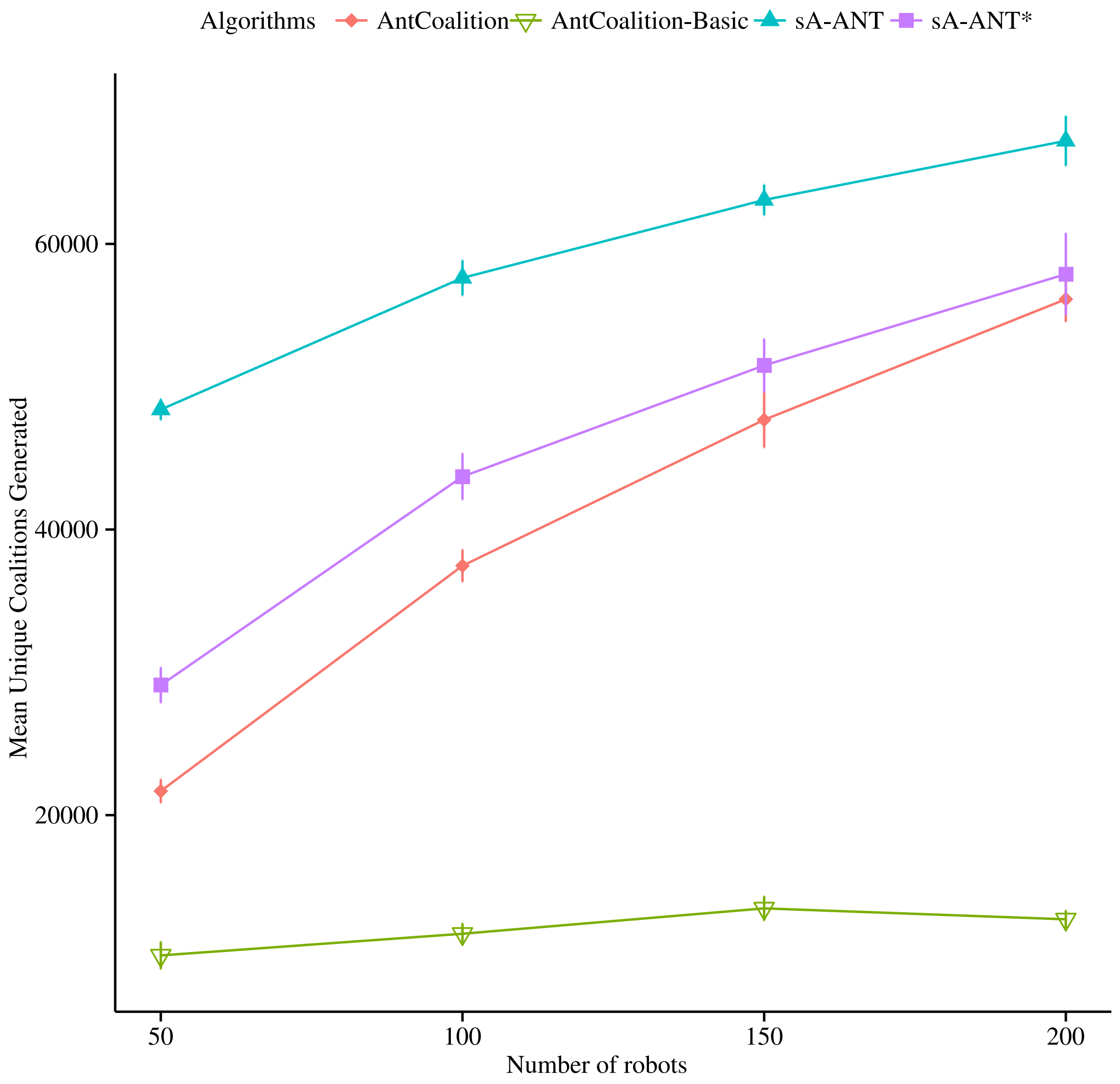


Figure III.6: The mean number of unique coalition generated by the algorithms with increasing number of robots.

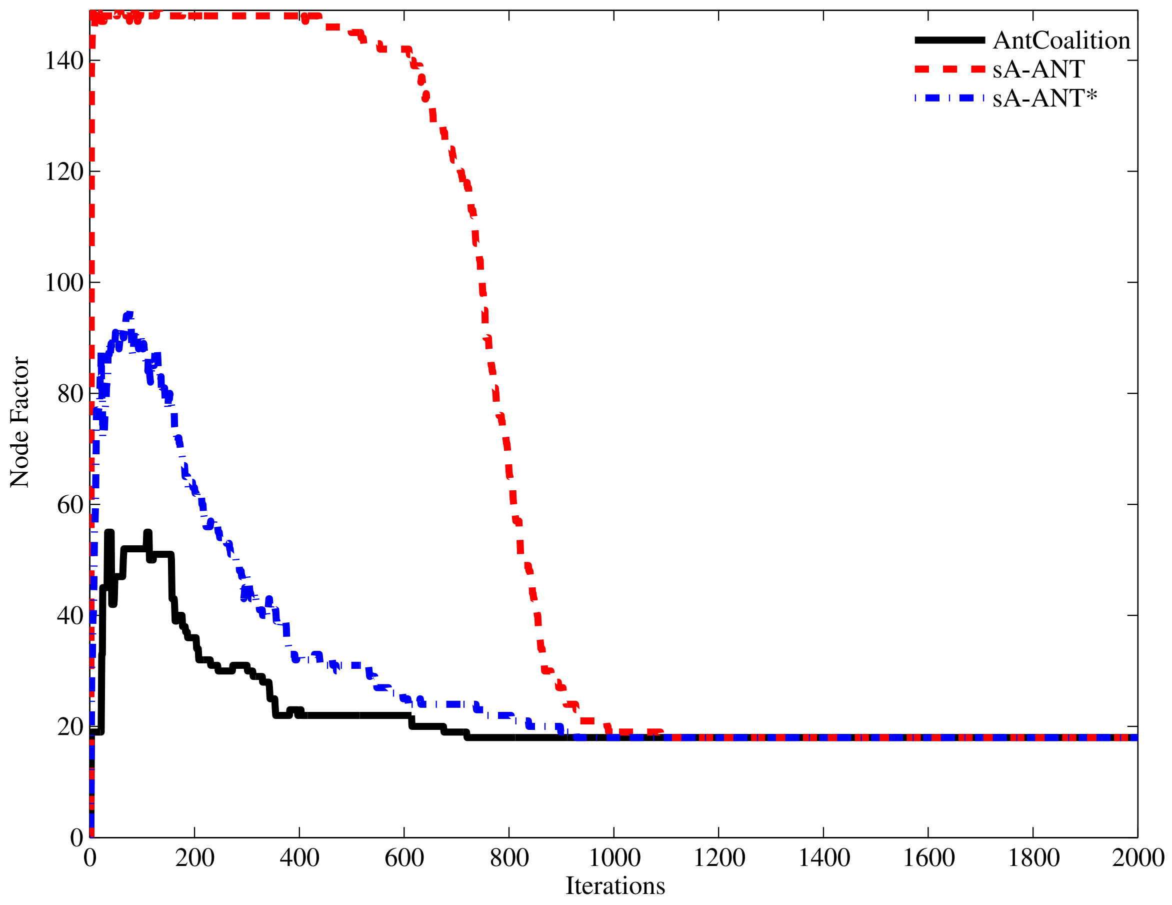


Figure III.7: The total node factor with increasing iterations of the algorithms for 150 robots ($\lambda = 0.04$).

III.4.2.1.1.1 *Node factor* measure

The significantly higher number of unique coalitions generated by *sA-ANT* and *sA-ANT** for the coalition formation problem is due to the extensiveness of the hybrid algorithms' search capability. An explicit measure of this search exploration is each algorithms' total node factor, which is calculated based on the average branching factor metric used in the TSP experiments. All the ACO algorithms for the coalition formation problem utilize the node pheromone depositing model; therefore, the node factor ($node\_factor(r)$) is defined as:

$$node\_factor(r)_\lambda = \sum_{k=1}^{|\mathcal{N}^r|} \tau_k^r > (min(\tau_r) + \lambda \times (max(\tau_r) - min(\tau_r)), \tag{III.8}$$

where $\lambda \in \{0.04, 0.06, 0.08\}$, τ_k^r denotes the pheromone amount on node $k \in \mathcal{N}^r$ that is connected to r, $\mathcal{N}^r$ represents the set of neighboring nodes of r, while $min(\tau_r)$ and $max(\tau_r)$ denote the minimum and maximum pheromone values of all the nodes that are connected to node r. The total node factor is computed as $\sum_{r=1}^{n} node\_factor(r)_\lambda$, where n is the number of agents in the system.

Figures III.7 and III.8 illustrate the higher searching capability of *sA-ANT** and *sA-ANT*, as compared to *Ant-Coalition*, which leverages only the single best ant to deposit pheromones. The total node branching factor for *Ant-Coalition-Basic* is not discussed, because unlike the remaining three algorithms, *Ant-Coalition-Basic* does not bound the pheromones; therefore, the algorithm experiences an exponential evaporation of node pheromone levels. Owing to the high number of ants responsible for depositing pheromones during the initial phases of the algorithm, *sA-ANT* maintains a very high total node factor. However, with a gradual decrease in the annealing temperature, the node factor for *sA-ANT* is lowered. Both *sA-ANT** and *sA-ANT* explore a greater search area, as compared to the *Ant-Coalition* algorithm, which is captured by their higher total node factor. Due to this enhanced searching, the hybrid algorithms generated a significantly higher number of unique coalitions, as compared to *Ant-Coalition* and *Ant-Coalition-Basic*. However, all the algorithms converged to the same number of nodes that denotes the final coalition size computed by each of the algorithm[3].

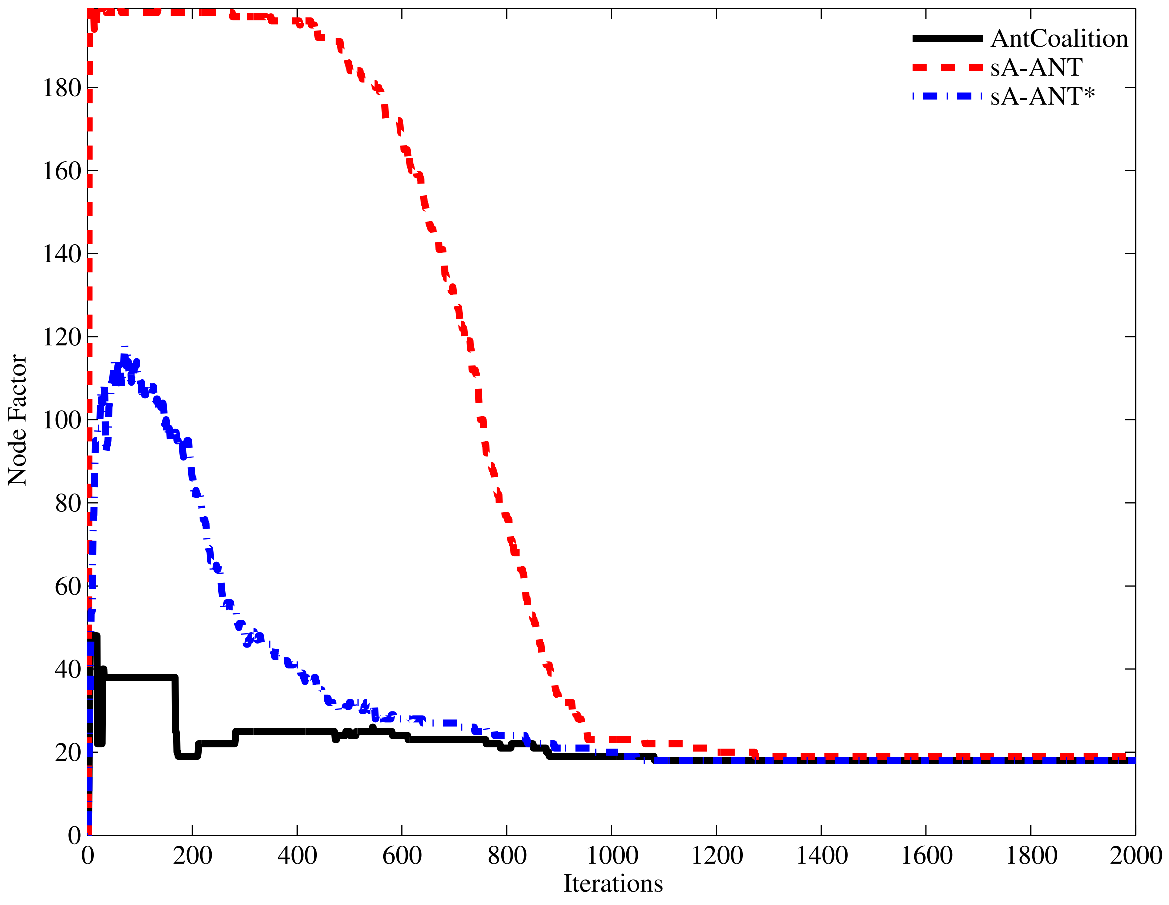


Figure III.8: The total node factor with increasing iterations of the algorithms for 200 robots ($\lambda = 0.04$).

[3]A same coalition size does not indicate the same coalition

III.4.2.1.2 Coalition Utility

The *coalition utility* metric demonstrates the quality of the generated coalitions. Table III.4 provides the mean coalition utilities for the algorithms averaged over all mission scenarios and trials for increasing n values. Both *sA-ANT** and *sA-ANT* generated coalitions of higher utility for all values of n, when compared to *Ant-Coalition* and *Ant-Coalition-Basic*. A Kruskal-Wallis one-way analysis of variance determined that there is a significant main effect of algorithm on coalition utility ($\chi^2 = 57.4$, $df = 3$, $p < 0.01$). Individual Wilcoxon rank-sum test comparisons were performed and the results are provided in Table III.5. Compared to *Ant-Coalition*, *sA-ANT** and *sA-ANT* generated significantly higher utility coalitions for $n = 150, 200$. However, for $n = 50, 100$, the improvement of the coalition utilities for *sA-ANT** and *sA-ANT* was not significant. When compared to *Ant-Coalition-Basic*, both *sA-ANT** and *sA-ANT* generated significantly higher utility coalitions for all values of n.

Number of Robots	**sA-ANT***	**sA-ANT**	**Ant-Coalition**	**AntCoalition-Basic**
50	**810374.5** (267368.2)	809937.2 (267025)	809863.6 (266950.4)	794579.7 (257218.1)
100	**852925.8** (298048.8)	852477.4 (297425.3)	851856.3 (297711.4)	831905.1 (284525.4)
150	**868749.9** (310622.1)	868401.0 (310549.6)	867204.6 (309878.2)	845189.4 (296827.6)
200	**876508.8** (314720.2)	876351.7 (314608.9)	874357.4 (313277.7)	853228.0 (300436.0)

Table III.4: The mean coalition utility of the algorithms for the coalition formation problem. The standard deviations are provided in parenthesis and the best solutions are highlighted in bold.

Number of Robots	**sA-ANT*** vs **Ant-Coalition**	**sA-ANT** vs **Ant-Coalition**	**sA-ANT*** vs **AntCoalition-Basic**	**sA-ANT** vs **AntCoalition-Basic**
50	$W = 301.5$, $z = -0.3$, $p = 0.4$	$W = 315.5$, $z = -0.07$, $p = 0.48$	$W = 72$, $z = -5.1$, $\mathbf{p < 0.01}$	$W = 75$, $z = -4.9$, $\mathbf{p < 0.01}$
100	$W = 261$, $z = -1.1$, $p = 0.16$	$W = 281$, $z = -0.6$, $p = 0.26$	$W = 0$, $z = -7.6$, $\mathbf{p < 0.01}$	$W = 0$, $z = -7.68$, $\mathbf{p < 0.01}$
150	$W = 231$, $z = -1.6$, $\mathbf{p = 0.031}$	$W = 245$, $z = -1.3$, $\mathbf{p = 0.038}$	$W = 0$, $z = -7.6$, $\mathbf{p < 0.01}$	$W = 0$, $z = -7.68$, $\mathbf{p < 0.01}$
200	$W = 181$, $z = -2.5$, $\mathbf{p < 0.01}$	$W = 196$, $z = -2.26$, $\mathbf{p < 0.01}$	$W = 0$, $z = -7.68$, $\mathbf{p < 0.01}$	$W = 0$, $z = -7.68$, $\mathbf{p < 0.01}$

Table III.5: Wilcoxon Rank Sum test results for comparing coalition utilities generated by the algorithms. $p < 0.05$ is significant, while $p < 0.01$ is highly significant.

III.4.2.1.3 Total Travel Distance

The *robot traveling distance* metric measures the total traversal distance for the coalition's robots to navigate to the task's location. Figure III.9 shows the mean traveling distance for the algorithms' coalition members averaged over mission scenarios and trials for all n values. Both *sA-ANT* and *sA-ANT** produced higher utility solutions with significantly smaller traveling distances for all values of n. *sA-ANT** generated higher quality solutions requiring even lower traveling cost when $n = 200$, as compared to *sA-ANT*. A Kruskal-Wallis one-way analysis of variance determined that there is a significant main effect for the algorithms based on the travel distance ($\chi^2 = 57.49,\ df = 3,\ p < 0.01$). A Wilcoxon rank-sum test was conducted for individual comparisons and the results are delineated in Table III.6. Both *sA-ANT* and *sA-ANT** resulted in coalitions that required significantly shorter traveling distances to the task location when compared to the *Ant-Coalition* and *Ant-Coalition-Basic* algorithms for all values of n.

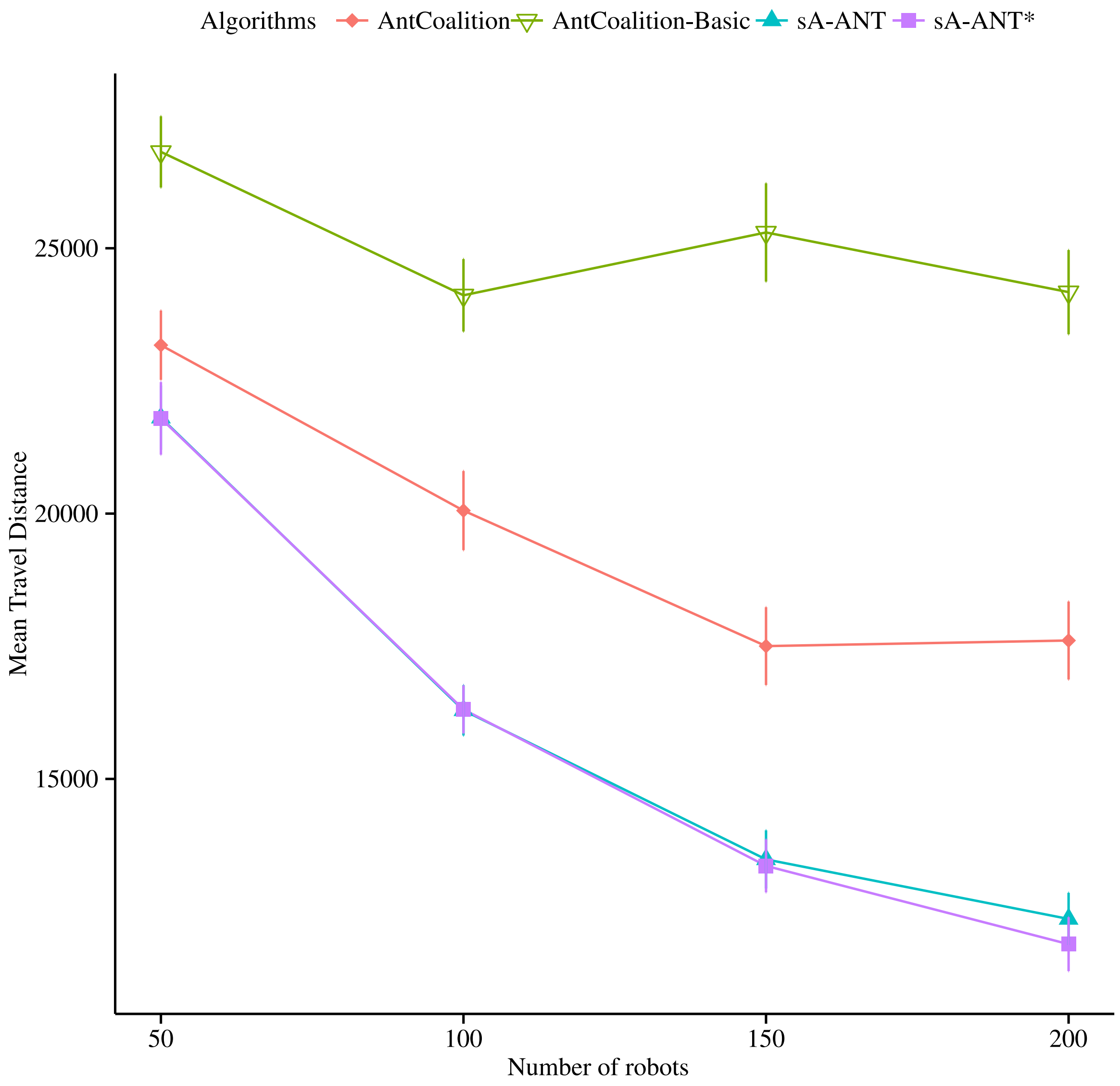


Figure III.9: The mean traveling cost for the missions with increasing number of robots.

Number of Robots	**sA-ANT*** vs **Ant-Coalition**	**sA-ANT** vs **Ant-Coalition**	**sA-ANT*** vs **AntCoalition-Basic**	**sA-ANT** vs **AntCoalition-Basic**
50	$W = 250$, $z = -1.33$, $\mathbf{p < 0.01}$	$W = 250$, $z = -1.3$, $\mathbf{p < 0.01}$	$W = 253$, $z = -4.23$, $\mathbf{p < 0.01}$	$W = 252$, $z = -4.2$, $\mathbf{p < 0.01}$
100	$W = 340$, $z = -3.9$, $\mathbf{p < 0.01}$	$W = 337$, $z = -3.8$, $\mathbf{p < 0.01}$	$W = 394$, $z = -6.24$, $\mathbf{p < 0.01}$	$W = 393$, $z = -6.21$, $\mathbf{p < 0.01}$
150	$W = 337$, $z = -3.9$, $\mathbf{p < 0.01}$	$W = 330$, $z = -3.6$, $\mathbf{p < 0.01}$	$W = 399$, $z = -6.6$, $\mathbf{p < 0.01}$	$W = 397$, $z = -6.5$, $\mathbf{p < 0.01}$
200	$W = 368$, $z = -4.9$, $\mathbf{p < 0.01}$	$W = 359$, $z = -4.76$, $\mathbf{p < 0.01}$	$W = 400$, $z = -6.75$, $\mathbf{p < 0.01}$	$W = 386$, $z = -6.2$, $\mathbf{p < 0.01}$

Table III.6: Wilcoxon Rank-Sum test results for traveling distances for coalition members computed by the algorithms. $p < 0.05$ is significant, while $p < 0.01$ is highly significant.

III.4.2.1.4 Computation Time

The time to calculate the robot coalitions is measured by the *computation time* metric. Figure III.10 provides the computational time of each algorithm as n increases, which increases for all the algorithms. A Kruskal-Wallis one-way analysis of variance determined that there is a significant main effect of algorithm on the computation time ($\chi^2 = 79.9$, $df = 3$, $p < 0.01$). A Wilcoxon rank-sum test was conducted for individual comparisons and the results are provided in Table III.7. The computational time of *sA-ANT* is significantly greater than that of *sA-ANT**, because *sA-ANT* generates a significantly higher number of unique coalitions. Moreover, the computational times of both the variants of *sA-ANT* are significantly higher than that of the *Ant-Coalition* algorithm. *Ant-Coalition* leverages only the single global best coalition for pheromone update; therefore, requiring lower computational time. The *Ant-Coalition-Basic* algorithm uses only ten ants during the iterations compared to fifty ants for the other three algorithms, thus its computational time is significantly lower.

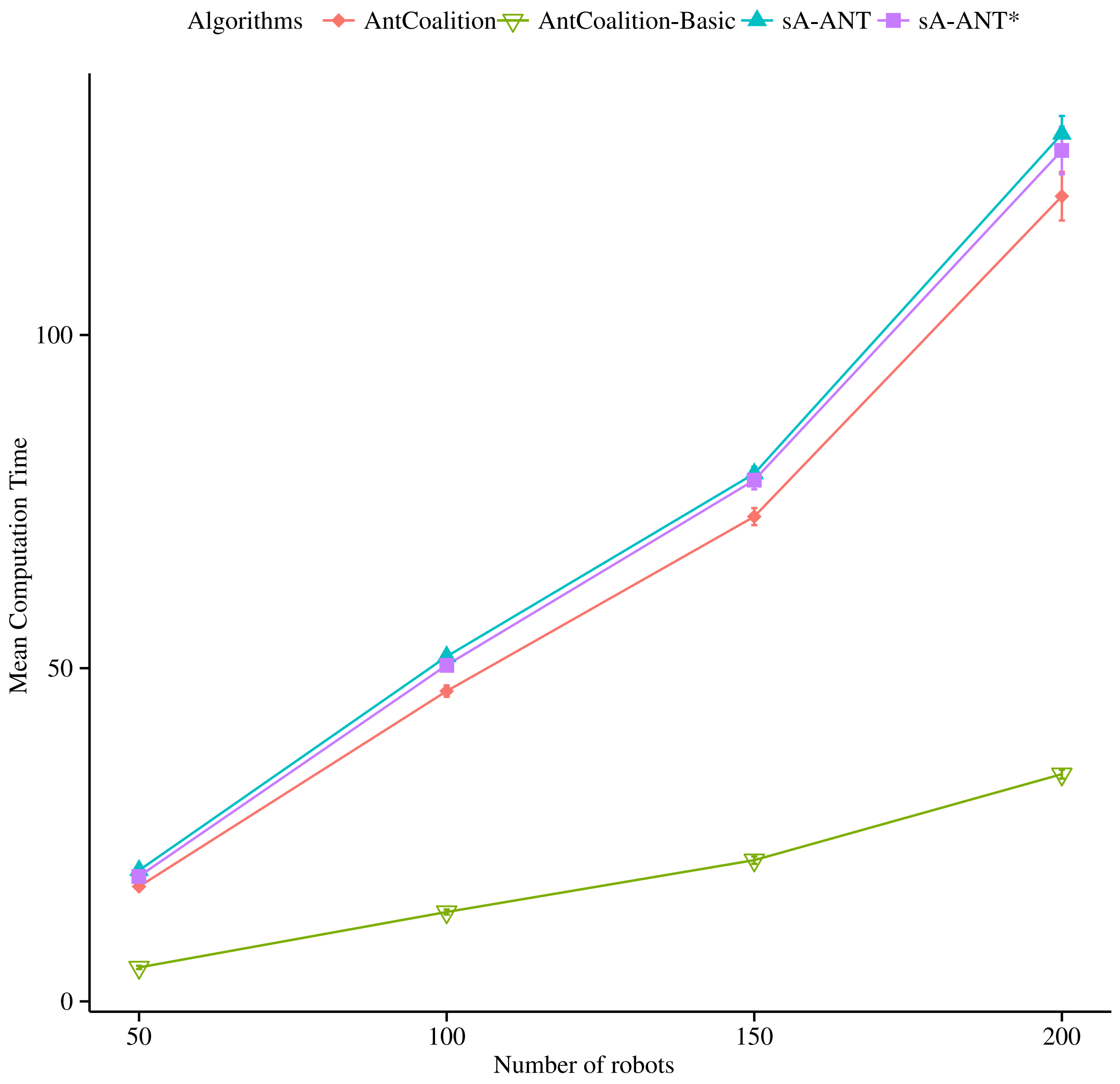


Figure III.10: The mean computational time (in seconds) with increasing number of robots.

Number of Robots	**sA-ANT*** vs **sA-ANT**	**sA-ANT*** vs **Ant-Coalition**	**sA-ANT** vs **Ant-Coalition**	**sA-ANT*** vs **AntCoalition-Basic**	**sA-ANT** vs **AntCoalition-Basic**
50	$W = 583$, $z = -5.24$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.06$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.06$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.05$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.05$, $\mathbf{p < 0.01}$
100	$W = 540.5$, $z = -4.42$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.05$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.05$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.06$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.06$, $\mathbf{p < 0.01}$
150	$W = 460.5$, $z = -2.8$, $\mathbf{p < 0.01}$	$W = 0.5$, $z = -6.0$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.0$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.0$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.0$, $\mathbf{p < 0.01}$
200	$W = 455$, $z = -2.75$, $\mathbf{p < 0.01}$	$W = 47$, $z = -5.1$, $\mathbf{p < 0.01}$	$W = 23$, $z = -5.6$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.0$, $\mathbf{p < 0.01}$	$W = 0$, $z = -6.0$, $\mathbf{p < 0.01}$

Table III.7: Wilcoxon Rank-Sum test results for comparing the computation time of the algorithms.

III.4.2.2 Discussion of coalition formation results

The *sA-ANT* and *sA-ANT** algorithms were tailored as greedy coalition formation algorithms that can generate high quality coalitions with the least traveling cost for the coalition members in a multi-agent settings. Both algorithms provide improved searching capability, as demonstrated by the significantly higher number of unique coalition generated, which supports hypothesis H_3. The qualities of the coalitions generated by *sA-ANT* and *sA-ANT** were significantly higher than that by *Ant-Coalition* for $n = 150, 200$ and those by *Ant-Coalition-Basic* for all n values; therefore, supporting hypothesis H_4. The traveling distances of the coalition members, as computed by *sA-ANT* and *sA-ANT** were significantly shorter than that of the other two algorithms; thereby, supporting hypothesis H_5. *sA-ANT* leverages a large number of ants to deposit pheromones; therefore, requiring the highest computational time. Despite using a single best fit ant for pheromone depositing, *sA-ANT** maintains and modifies a dynamically modulated repository of solutions; thereby, requiring significantly higher computational time, as compared to *Ant-Coalition* and *Ant-Coalition-Basic* as hypothesized in H_6. However, all the algorithms are very scalable when it comes to large scale teams of robots. The mean computational time to generate a coalition for a team of 200 robots was 130 seconds for *sA-ANT*, 127 seconds for *sA-ANT**, and 120 seconds for *Ant-Coalition*.

III.4.3 Experimental Design for the Maximal Clique Problem

The *maximal clique problem* experiment involved comparing the performance of *sA-ANT** and *sA-ANT* to the *Ant-Clique* algorithm (Solnon and Fenet, 2006), which was implemented in C++ using Qt. The *Maximal Clique* maximization problem seeks to generate cliques of a maximum size. The ***DIMACS BENCHMARK***

LIBRARY (Mascia, 2014) was used for the experiments. The benchmark library contains a number of instances of the *Maximal Clique* problem and their respective optimal maximal clique sizes. A total of fifteen problem instances were selected. Four problem instances (*C125.9*, *C250.9*, *C500.9*, *C1000.9*), with graphs containing 125 to 1000 nodes are defined in the format "*CXXX.9*", where XXX indicates the number of nodes in the undirected graphs and 0.9 denotes the edge density of each graph. Six other instances (*brock200_2*, *brock200_4*, *brock400_2*, *brock400_4*, *brock800_2*, *brock800_4*), with node counts ranging from 200 to 800 have the format "*brockXXX_Y*", where XXX denotes the number of nodes in the graph, and Y is either 2 or 4. Finally, the remaining five instances (*gen200_p0.9_44*, *gen200_p0.9_55*, *gen400_p0.9_55*, *gen400_p0.9_65*, *gen400_p0.9_75*), with node counts of 200 and 400 are in the format "*genXXX_p0.9_ZZ*", where XXX represents the number of graph nodes, $p0.9$ indicates an edge density of 0.9 for each graph, and ZZ denotes the optimal clique size for the respective instance. The performance of *sA-ANT** and *sA-ANT* was compared to the *Ant-Clique* algorithm (Solnon and Fenet, 2006), the only existing ACO-based clique solver. The *Ant-Clique* parameter values were set to $\alpha = 1$, $\rho = 0.99$, $m = 30$, $\tau_{max} = 6.0$, $\tau_{min} = 0.01$, as defined in its original implementation (Solnon and Fenet, 2006). The parameter settings for *sA-ANT** and *sA-ANT* were $\alpha = 1$, $\rho = 0.99$, $m = 30$, the same as that of the *Ant-Clique*. The upper bound for the edge pheromones was set as $\tau_{max} = \frac{1}{(1-\rho)\times(1+|C_{gBest}|-|C_{iterBest}|)}$ (C_{gBest} and $C_{iterBest}$ represent the current global best and iteration best clique sizes, respectively), as theoretically proved for any pheromone bounded ACO approach (Stutzle and Hoos, 2000). Twenty five trials, each consisting 10,000 iterations were conducted for each algorithm. The associated hypotheses are:

- H_7 : The search exploration capability of *sA-ANT** and *sA-ANT* will be greater than that of *Ant-Clique*.
- H_8 : The maximal clique sizes computed by *sA-ANT** and *sA-ANT* will be significantly greater than that generated by *Ant-Clique*.

III.4.3.1 Maximal Clique Results

The performance of *sA-ANT**, *sA-ANT*, and *Ant-Clique* was compared based on the extensiveness of their search capability and the mean sizes of the computed maximal cliques.

III.4.3.1.1 Search Exploration

The search exploration capability of the algorithms (*sA-ANT**, *sA-ANT*, and *Ant-Clique*) was measured using the *number of active edges* metric ($E_{actv}(\lambda)$), which is defined as

$$E_{actv}(\lambda) = \sum_{r=1}^{n} \sum_{k=1}^{|\mathscr{N}^r|} \tau_{rk} > (min(\tau_r) + \lambda \times (max(\tau_r) - min(\tau_r)), \qquad \text{(III.9)}$$

where λ is a constant, n is the number of graph nodes, τ_{rk} denotes the pheromone amount on the edge that connects nodes r and $k \in \mathcal{N}^r$, $\mathcal{N}^r$ represents the set of neighboring nodes of r, while $min(\tau_r)$ and $max(\tau_r)$ denote the minimum and maximum pheromone values of all the edges that are incident on node r. The maximum value of $E_{actv}(\lambda)$ for a given problem instance is $2 \times e$, where e is the number of edges in the clique graph.

Each algorithm begins by initializing every edge pheromone to τ_0. The pheromone levels of the edges belonging to good solutions are increased via positive reinforcements by depositing additional pheromones as each algorithm progresses. Moreover, due to the exponential evaporation that occurs during the algorithm iterations, the pheromone amounts of several graph edges decrease when they are not reinforced. The edges with higher pheromones attract more ants during the search process; thereby, facilitating a greater exploration of the search space. The *number of active edges* metric accounts for the number of graph edges with pheromone values greater than a particular threshold that continue to remain active for further exploration by the ant colony.

A high value of $E_{actv}(\lambda)$ signifies a high fraction of the total number of graph edges that have been explored by the ants during the search process. Figures III.11, III.12, and III.13 illustrate the *number of active edges* for each algorithm. *sA-ANT* exhibits the highest *number of active edges*, because it employs a much higher number of ants to deposit pheromones during the early algorithm phases. Compared to *sA-ANT*, *sA-ANT** and *Ant-Clique* leverage only a single ant for pheromone depositing; therefore, each results in a lower value of $E_{actv}(\lambda)$. Despite this single ant depositing policy, *sA-ANT** exhibits higher search exploration than the *Ant-Clique* due to the diversity of solutions in its repository during the initial iterations, thus *sA-ANT** generates greater $E_{actv}(\lambda)$ values.

For example, consider the $C125.9$ problem instance that contains 125 nodes and 6963 edges (e) with an edge density of 0.9. The maximum value of $E_{actv}(\lambda) = 2 \times e$, because each edge can be counted at most twice for its respective nodes; therefore, $E_{actv}^{max}(\lambda) = 2 \times 6963 = 13926$. Figure III.11 demonstrates that the *number of active edges* peaked to 13200 very rapidly during the initial iterations for *sA-ANT*. *sA-ANT** and *Ant-Clique* achieved $E_{actv}^{max}(\lambda) = 9139$ and $E_{actv}^{max}(\lambda) = 8955$, respectively. Both *sA-ANT* and *sA-ANT** maintain greater search exploration for a larger number of iterations before converging to the final maximal clique solution. A more aggressive search exploration example is observed for instances $C250.9$ and $gen400\_p0.9\_55$, as shown in Figures III.12 and III.13, respectively. The remaining problem instances demonstrated a similar trend for $E_{actv}(\lambda)$.

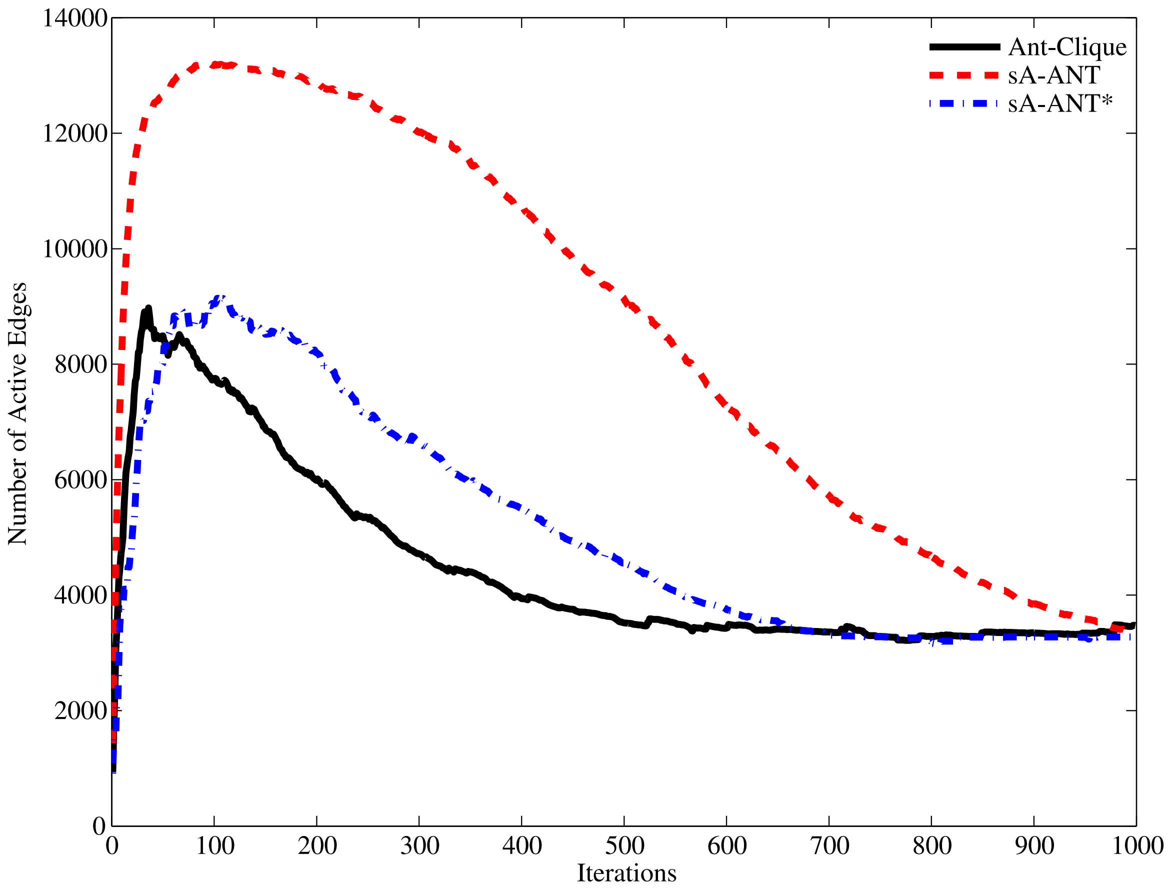


Figure III.11: The change in the *number of active edges* for *sA-ANT**, *sA-ANT*, and *Ant-Clique* for problem instance $C125.9$. The number of edges in graph is 6963 and $E_{actv}^{max}(\lambda) = 13926$.

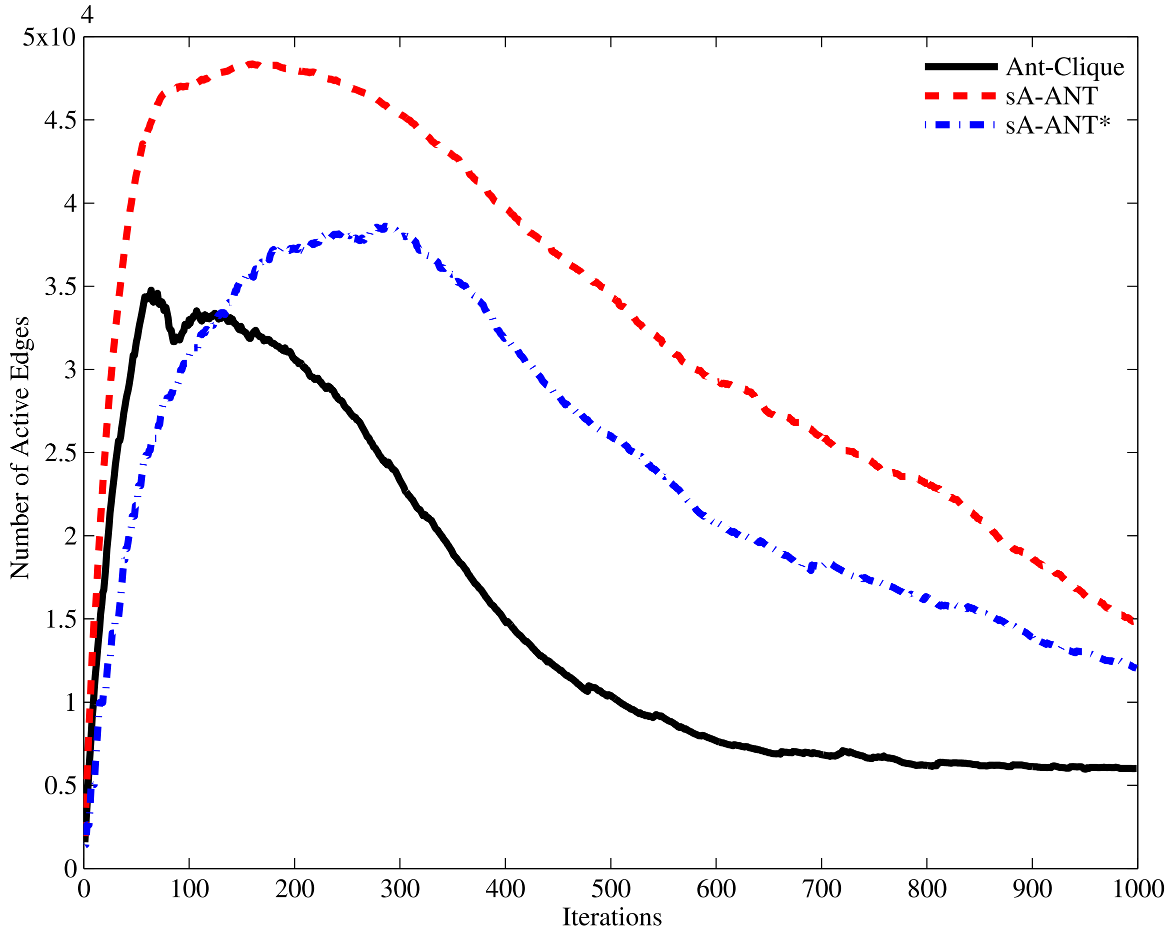


Figure III.12: The change in the *number of active edges* for *sA-ANT**, *sA-ANT*, and *Ant-Clique* for problem instance *C*250.9. The number of edges in graph is 27984 and $E_{actv}^{max}(\lambda) = 55968$.

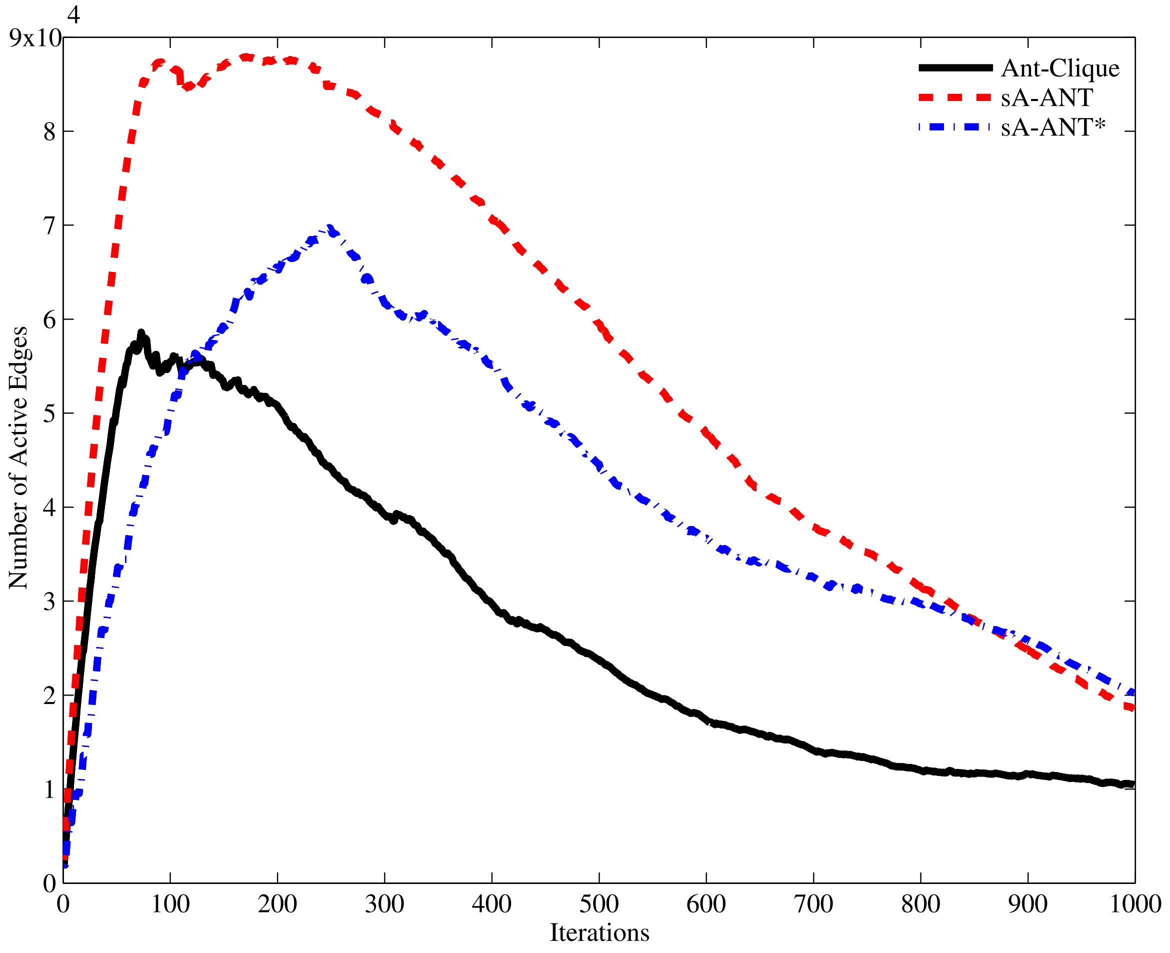


Figure III.13: The change in the *number of active edges* for *sA-ANT**, *sA-ANT*, and *Ant-Clique* for problem instance *gen*400_*p*0.9_55. The number of edges in graph is 71820 and $E_{actv}^{max}(\lambda) = 143640$.

III.4.3.1.2 Clique Sizes

The best and the mean maximal cliques obtained by the three algorithms for each of the fifteen clique instances are highlighted in Table III.8, which also highlights the frequency, or the number of times each algorithm achieved its best maximal clique size. The results demonstrate that both hybrid algorithms performed better than *Ant-Clique* on six problem instances (*C500.9*, *C1000.9*, *brock200_4*, *brock400_4*, *brock800_2*, *brock800_4*), and performed identically on six other instances (*C125.9*, *C250.9*, *brock200_2*, *gen200_p0.9_55*, *gen400_p0.9_65*, *gen400_p0.9_75*), where all algorithms generated the optimal clique sizes over 25 trials. *sA-ANT** generated maximal cliques with the largest mean sizes on five instances (*C500.9*, *C1000.9*, *brock200_4*, *brock400_4*, *gen400_p0.9_55*), as highlighted in Table III.8. Moreover, *sA-ANT**'s frequency in attaining its best solution is high when compared to *sA-ANT* and *Ant-Clique*, as demonstrated for instances: (1) *brock*200_4, (2) *brock*400_4, and (3) *gen*400_*p*0.9_55. A Kruskal-Wallis one-way analysis determined that there was no significant main effect of algorithm on the mean clique sizes. Individual Wilcoxon rank-sum

Problem Instances	Optimal	*sA-ANT** Best Found [freq.]	*sA-ANT** Mean (stdv.)	*sA-ANT* Best Found [freq.]	*sA-ANT* Mean (stdv.)	*Ant-Clique* Best Found [freq.]	*Ant-Clique* Mean (stdv.)
C125.9	34	34 [25]	**34 (0.0)**	34 [25]	**34 (0.0)**	34 [25]	**34 (0.0)**
C250.9	44	44 [25]	**44 (0.0)**	44 [25]	**44 (0.0)**	44 [25]	**44 (0.0)**
C500.9	>=57	**57 [2]**	**55.9 (0.4)**	56 [17]	55.7 (0.7)	57 [1]	55.6 (0.6)
C1000.9	>=68	**68 [1]**	**66.4 (0.8)**	67 [9]	66.2 (0.7)	67 [2]	65.9 (0.7)
brock200_2	12	12 [25]	**12 (0.0)**	12 [25]	**12 (0.0)**	12 [25]	**12 (0.0)**
brock200_4	17	**17 [25]**	**17 (0.0)**	17 [24]	16.96 (0.2)	17 [17]	16.7 (0.5)
brock400_2	29	29 [1]	24.6 (0.4)	25 [24]	24.97 (0.2)	**29 [2]**	**25.0 (1.2)**
brock400_4	33	**33 [10]**	**27.7 (4.3)**	33 [8]	27.1 (4.1)	33 [7]	26.7 (4.1)
brock800_2	24	21 [3]	19.9 (0.6)	**24 [1]**	**20.5 (0.9)**	21 [2]	19.8 (0.5)
brock800_4	26	21 [1]	19.8 (0.5)	**26 [1]**	**19.9 (1.3)**	21 [2]	19.7 (0.6)
gen200_p0.9_44	44	44 [25]	**44.0 (0.0)**	44 [20]	43.3 (1.6)	44 [22]	43.5 (1.3)
gen200_p0.9_55	55	55 [25]	**55 (0.0)**	55 [25]	**55 (0.0)**	55 [25]	**55 (0.0)**
gen400_p0.9_55	55	**53 [21]**	**52.8 (0.6)**	52 [6]	51.3 (0.4)	53 [7]	51.9 (0.9)
gen400_p0.9_65	65	65 [25]	**65 (0.0)**	65 [25]	**65 (0.0)**	65 [25]	**65 (0.0)**
gen400_p0.9_75	75	75 [25]	**75 (0.0)**	75 [25]	**75 (0.0)**	75 [25]	**75 (0.0)**

Table III.8: The Best and Mean clique sizes for the Maximal Clique problem as computed by *sA-ANT**, *sA-ANT*, and *Ant-Clique* for fifteen clique instances. The standard deviation (stdv.) and the frequency (freq.) are provided alongside the mean and best found clique sizes, respectively. The best solutions are highlighted in bold.

tests found no statistically significant differences in performance across the three algorithms for any of the graph sizes.

III.4.3.2 Discussion of maximal clique results

The experimental results demonstrate that both *sA-ANT** and *sA-ANT* provide a more extensive search capability by integrating the *simulated annealing* methodology, which supports hypothesis H_7. There was a small difference in the mean maximal clique sizes generated across the three algorithms, with *sA-ANT** and *sA-ANT* performing better than the *Ant-Clique*. Compared to the results of the *Ant-Clique* variant that integrated local search (Solnon and Fenet, 2006), *sA-ANT** generated slightly better solutions for five clique instances without requiring local search techniques. All three algorithms computed maximal cliques that are either the optimal solutions or extremely close to the optimal sizes. However, since there was no significant difference between the results, hypothesis H_8 was not supported.

III.4.4 Overall Discussion

The effectiveness of *sA-ANT* and *sA-ANT** in addressing the search stagnation shortcoming of the existing state-of-the-art ACO approaches has been illustrated. Both algorithms exhibit enhanced search exploration capabilities, without stagnating in a local optima. The initial high annealing temperature and the gradual annealing schedule contribute to an improved search exploration during the initial phases of the algorithm that is demonstrated by their higher average branching and node factors compared to *Max-Min Ant System*, the current best ACO approach. The enhanced search capability of *sA-ANT** and its better computational time render it more potent than *sA-ANT* with the former generating the best results for the three problems that were investigated. The application of *sA-ANT** and *sA-ANT* to the three diverse combinatorial optimization problems clearly illustrates that the presented approach is a generic search algorithm capable of computing better solutions when compared to existing state-of-the-art ACO algorithms.

III.5 Distributed sA-ANT*

Distributed intelligent systems that autonomously create robust coalitions of heterogeneous robots are gaining incredible impetus for future critical and real-world missions. Real robots in unstructured, dynamic real-world missions suffer from frequent communication interruptions, robot failures, and lack of global information. The presented *sA-ANT* and *sA-ANT** algorithms are centralized, and thus will not support real-world, distributed multi-robot systems. Centralized algorithm can suffer from brittleness, unresponsiveness to dynamic environments, high communication requirements, and unscalability. This section presents the distributed variant of the *sA-ANT** algorithm, called *d-sA-ANT** which exploits the advantages of both the ant colony optimization and the simulated annealing techniques. The distributed algorithm incorporates an information diffusion methodology in large scale networks in order to propagate crucial robot information across a time-varying communication topology.

III.5.1 System Design

The presented decentralized algorithm falls into the category of cooperative multicolony optimization, according to an existing taxonomy (Pedemonte et al., 2011). Each robot in the team acts as an independent processor and concurrently computes the most appropriate coalition of robots that can accomplish a given task. Additionally, each robot deploys its own set of ants to search the problem space semi-independently, i.e., each robot manages its own ant colony. Each robot publishes its local best solution to the neighboring robots, such that every other robot can update its estimated global best solution. Each robot consists of several attributes: (1) robot identifier, (2) a set of time-varying neighboring robots, denoted by Γ_i^t, (3) local and global coalition information, and (4) a colony of ants whose size is determined by the processing power of the robot's onboard processor.

Unstructured and dynamic mission domains may intermittently disrupt the continuous communication between the robots in multi-robot settings; therefore, the robots form a time-varying communication network represented by an undirected graph, $G_t(V, E(t))$. Each robot contains only local information that is comprised of its immediate neighbors connected via a communication network. The use of just the neighboring robot information for deriving coalitions will either lead to lower quality solutions, or no solutions at all due to each robot having a restricted perspective and lack of information. Thus, *d-sA-ANT** incorporates information propagation based only on neighboring robots. Each robot, $R_i \in R$ publishes its local knowledge of its neighboring robots and their respective state information (e.g., resources, position, engagement status) to every other robot in its immediate neighborhood by means of information packets. During information sharing with each neighboring robot, R_i waits for a certain timeout time ($\kappa(R_i)$), which accounts for any communication latency in real-world networks. Each robot is assumed to detect faults in itself (*endogenous fault detection* (Christensen et al., 2009)) and publishes its health status (e.g., faulty, working) to every neighboring robot; therefore, in case of any robot failure, its neighbors will be updated. This propagation of fault information via neighboring robots renders *d-sA-ANT** robust during robot failures. When a robot, R_i detects any change in the network, i.e., a robot moves out of its communication range or a new robot comes within its communication vicinity, it either deletes or adds the particular robot to its list of neighboring robots, respectively. During every such event, R_i immediately publishes this new information to its immediate neighbors and information propagation occurs. Based on the acquired information of the robots in the team, each robot computes its forwarding table using the Dijkstra's shortest path algorithm.

The information propagation for *d-sA-ANT** is illustrated using Figures III.14 and III.15. Consider a situation, where seven heterogeneous robots are connected in a communication topology, as shown in Figure III.14. Each robot maintains an adjacency list to capture the network. At time step $t = 0$, each robot's adjacency list is shown alongside. Based on each robot's $\kappa(R_i)$, the modified adjacency lists of every robot is shown in Figure III.15. Each robot propagates only its neighbors' information to every robot in its neighborhood, which in turn is used by every other robot to update its adjacency list. Due to the information propagation, every robot converges to the overall network structure.

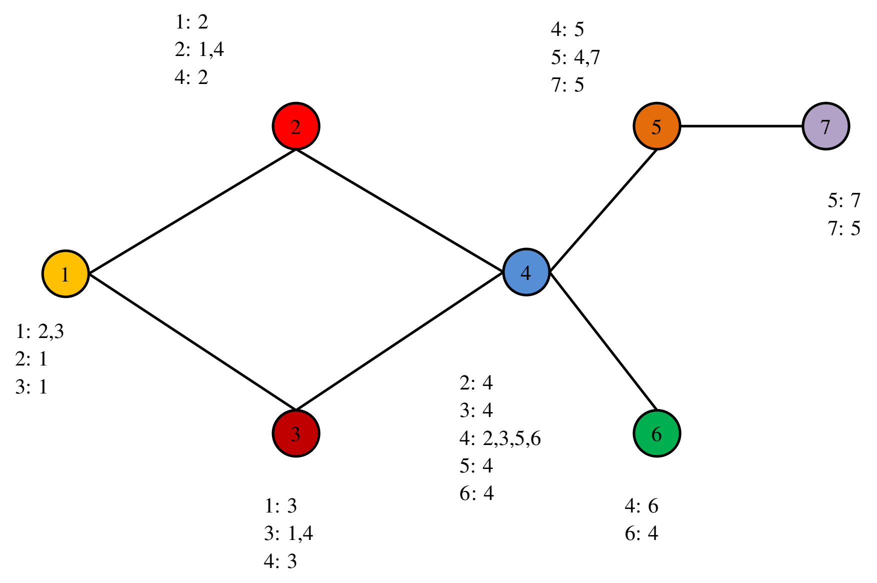


Figure III.14: Seven robots connected in a network topology. The adjacency list capturing the partial topology is provided alongside each robot node.

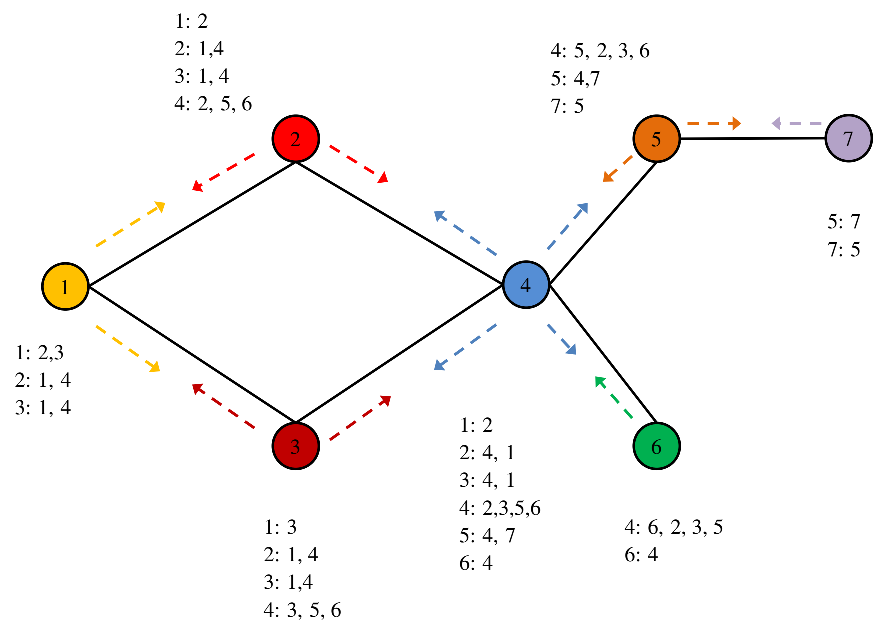


Figure III.15: Each robot propagates it own list to its neighbors as highlighted in their respective colors. Every robot updates its adjacency list accordingly.

The distributed *sA-ANT** algorithm is provided in Algorithm 3. Each robot deploys its own colony of ants to search the solution space in order to determine potential robot coalitions that can satisfy task requirements. The processing is very similar to that of the centralized *sA-ANT**; however, there are several key differences. At the start of every iteration, each robot, $R_i \in R$ propagates any updated information to its immediate neighbors in Γ_i^t (Line 4 in Algorithm 3). This step is crucial, because each robot maintains the most updated knowledge of the other robots and aids robustness, because any faulty robot can be removed during the coalition computation. After every iteration, each robot compares the individual solution of each ant and computes the local best coalition. Each ant then propagates its local best solution to its immediate neighbors, and each robot compares its current local best coalition with that of its neighboring robots. When a higher quality solution is realized, each robot updates its current global best solution. This updated global solution is leveraged by the ants of each robot to deposit and update the respective pheromone matrix. Once the pheromone update is performed, the ant colonies resume the computations until a global coalition is achieved. Once a coalition is formed, each member robot updates its task engagement status to *assigned* and publishes this information to its immediate neighbors. Every robot updates this information and does not consider the assigned robots for coalition calculations for additional tasks.

Algorithm 3 The *d-sA-ANT** algorithm, $\forall R_i \in R$

```
Input: Set of ACO parameters ($\alpha$, $\beta$, $\rho$); Set of simulated annealing parameters ($\gamma$, Annealing Temperature,
    anneal_temperature); Task $M_j \in M$; Number of iterations, nIter; Number of ants, nAnts
 1: $S_{local\_best} \leftarrow \emptyset$
 2: $S_{global\_best} \leftarrow \emptyset$
 3: for iteration=1 to nIter do
 4:     $PropagateInfo(R_i, \Gamma_i^t)$
 5:     for $\lambda$=1 to nAnts do
 6:         $randRobot \leftarrow rand(R \setminus (S_{c,\lambda} \cup faultyRobot)$
 7:         while $!isTaskFulfilled(M_j, S_{c,\lambda})$ do
 8:             $candidateRobot \leftarrow R'_j \in R \setminus (S_{c,\lambda} \cup faultyRobot)$
 9:             $S_{c,\lambda} \leftarrow S_{c,\lambda} \cup candidateRobot$
10:         end while
11:         Update $S_{local\_best}$
12:     end for
13:     $Publish(S_{local\_best}, \Gamma_i^t)$
14:     Update $S_{global\_best}$
15:     Update repository using $S_{local\_best}$ and $S_{global\_best}$
16:     Select $S*$ for local pheromone update
17:     $anneal\_temperature \leftarrow anneal\_temperature \times \gamma$
18: end for
19: Update robot engagement status
```

III.5.2 Experimental Design

The primary objective of the evaluation was to test the efficacy of *d-sA-ANT** when compared to its centralized variant, *sA-ANT**. The *d-sA-ANT** algorithm was implemented using the Qt framework that provided the QThreadpool for generating parallel threads to simulate each robot. Each thread leverages Qt's signal and slot mechanism to implement a publish-subscribe messaging framework via which the robots perform the information propagation with their respective neighbors. The number of robots was set to $n = 20$ and 50. A total of seven mission scenarios were used and were randomly generated, each comprised of a single task requiring five resource types. The resource requirements were fixed for a particular mission across all trials. The first mission scenario began with each resource type value set to ten. Each additional scenario increased the resource type values by five, to a maximum of forty for each task requirement value during the final mission. The communication range for each robot was set to 100m, 250m, and 1000m based on the existing wireless communication protocols IEEE 802.11.a, IEEE 802.11.n, and IEEE 802.11y, respectively. Each robot, based on the selected communication range, thus had a restricted ability to communicate with all other robots. The robots that fall into the communication range of a particular robot are considered to be the latter's immediate neighbors. The robots are instantiated at random locations in the environment, and remained stationary during each trial during the computation of the coalitions. Each robot propagates messages that include its resource capabilities, location, and immediate neighbor information.

Up to twenty trials were run for each of the seven mission scenarios for each communication range and number of robots until ten successful coalition allocations were collected. Each trial for a particular mission and n value used the same task requirements. The robot locations, task locations and robot capabilities were randomly generated for a particular mission trial and n value, but remained the same across algorithms. The task and robot locations were generated to lie within a 1000m x 1000m area. The task utility, or reward was set as $U(T_j) = 5000 \times \sum_{i=1}^{5} TR_{ji}$ in order to maintain uniformity across all missions, trials, and algorithms. Three hypotheses were analyzed during the experiments:

- H_9 *d-sA-ANT**'s computation time will be significantly higher than that of *sA-ANT**.
- H_{10}: *d-sA-ANT** will generate coalitions of similar utility as that computed by *sA-ANT**.
- H_{11}: The traveling distance of the members of the coalitions generated by *d-sA-ANT** and *sA-ANT** will be similar.

III.5.2.1 Experimental Results

A total of 20 trials were performed with the communication range of 100m for $n = 20$ and 50. Coalitions were obtained during only 3 out of 20 trials with $n = 20$ for the first mission scenario, where the resource requirements were set to ten for every resource type. Out of the 20 trials, for $n = 50$, only seven trials resulted in coalitions for the first mission with resource requirements set to ten units. None of the 20 trials for any of the remaining six missions for either value of n generated any coalitions. The 100m communication range in an arena of size 1000m $\times$ 1000m resulted in no robot clusters (coalitions) that were connected via the communication topology having the necessary resources to meet the task requirements.

		Communication Range (m)	
		250	1000
Number of	20	11	14
Robots	50	16	18

Table III.9: Number of trials of out 20 that resulted in coalitions for the 250m and 1000m communication ranges with *d-sA-ANT**.

Both *sA-ANT** and *d-sA-ANT** generated coalitions when the communication range was increased to 250m and 1000m. Table III.9 provides the number of successful trials out of twenty for which coalitions were obtained, given the number of robots and communication ranges. The ten trials with the highest coalition utilities were used for the analysis presented in this section. Four metrics were used to compare the coalitions generated by the algorithms: (1) Computation time, (2) Coalition utility, (3) Traveling distance, and (4) Coalition size and composition.

III.5.2.1.1 Computation time

The coalition generation computation time using the algorithms when $n = 20$ with the 250m and 1000m communications ranges are provided in Figure III.16. The same comparisons for $n = 50$ are presented in Figure III.17. Generally speaking, *d-sA-ANT** has higher computation time than *sA-ANT** for a particular communication range, independent of the number of robots.

A Kruskal-Wallis one-way analysis of variance determined that there is a significant main effect of algorithm on the computational time for both communication ranges for $n = 20$ ($\chi^2 = 6.3, df = 1, p < 0.01$). Wilcoxon rank-sum tests were conducted for individual comparisons and the results are provided in Table III.10. *d-sA-ANT** required significantly higher computational time, compared to *sA-ANT** for $n = 20$ and the 250m communication range for all missions. The computation time of *d-sA-ANT** was significantly higher than *sA-ANT** with communication range of 1000m for six mission scenarios. The general trend in the computation time for *d-sA-ANT** and *sA-ANT** is that both increase with the increase in the number of task requirements across the mission scenarios.

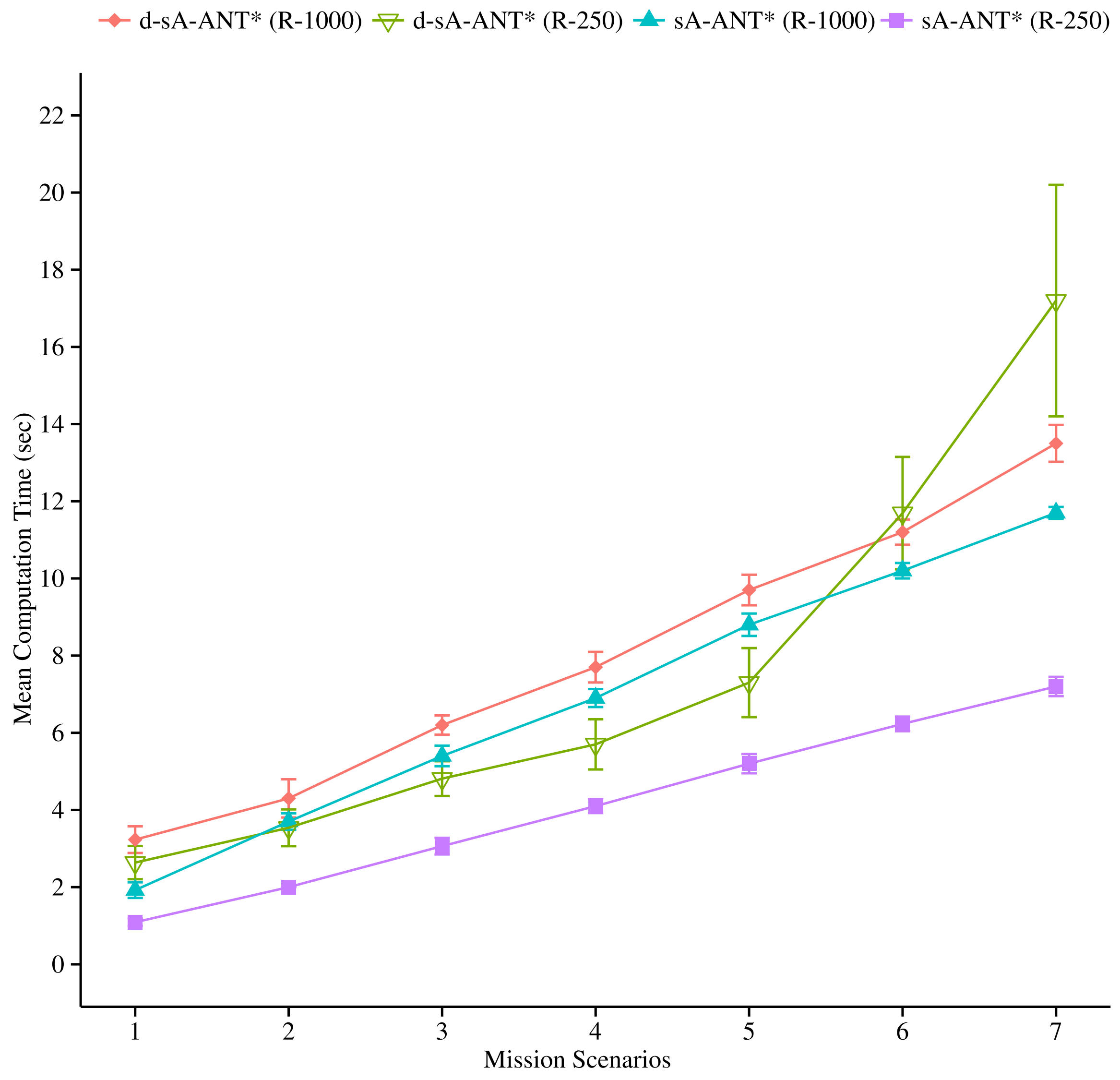


Figure III.16: The mean computation time for coalition formation for both the 250m and 1000m communication ranges and 20 robots.

Mission Scenarios	Comm. Range = 250m **sA-ANT*** vs **d-sA-ANT***	Comm. Range = 1000m **sA-ANT*** vs **d-sA-ANT***
1	$W = 101.5$, $z = -3.0$, $\mathbf{p < 0.01}$	$W = 138$, $z = -2.9$, $\mathbf{p < 0.01}$
2	$W = 149.5$, $z = -3.6$, $\mathbf{p < 0.01}$	$W = 57$, $z = -0.5$, $p = 0.29$
3	$W = 164.5$, $z = -3.3$, $\mathbf{p < 0.01}$	$W = 74$, $z = -1.9$, $\mathbf{p = 0.02}$
4	$W = 83.5$, $z = -2.5$, $\mathbf{p < 0.01}$	$W = 72.3$, $z = -1.8$, $\mathbf{p = 0.03}$
5	$W = 73$, $z = -2.2$, $\mathbf{p < 0.01}$	$W = 103.5$, $z = -2.4$, $\mathbf{p = 0.018}$
6	$W = 143$, $z = -3.0$, $\mathbf{p < 0.01}$	$W = 77.5$, $z = -2.1$, $\mathbf{p < 0.01}$
7	$W = 90.5$, $z = -3.1$, $\mathbf{p < 0.01}$	$W = 88.5$, $z = -2.9$, $\mathbf{p < 0.01}$

Table III.10: Wilcoxon Rank Sum test results comparing computation time of the algorithms for the two communication ranges and 20 robots.

The computation times of *d-sA-ANT** and *sA-ANT** for $n = 50$ are analyzed. A Kruskal-Wallis one-way analysis of variance determined that there is a significant main effect of algorithm on the computational time for both the communication ranges when $n = 50$ ($\chi^2 = 15.12, df = 1, p < 0.01$). Individual comparisons were performed using the Wilcoxon rank-sum test and the results are provided in Table III.11. The computation time of *d-sA-ANT** was significantly higher than that of *sA-ANT** for both the communication ranges across all the mission scenarios. The computation times of *d-sA-ANT** and *sA-ANT** for the communication ranges of 250m to 1000m are provided in Figure III.17.

Mission Scenarios	Comm. Range = 250m **sA-ANT*** vs **d-sA-ANT***	Comm. Range = 1000m **sA-ANT*** vs **d-sA-ANT***
1	$W = 49$, $z = -3.1$, $\mathbf{p < 0.01}$	$W = 100$, $z = -3.8$, $\mathbf{p < 0.01}$
2	$W = 64$, $z = -3.3$, $\mathbf{p < 0.01}$	$W = 100$, $z = -3.8$, $\mathbf{p < 0.01}$
3	$W = 100$, $z = -3.8$, $\mathbf{p < 0.01}$	$W = 256$, $z = -4.8$, $\mathbf{p < 0.01}$
4	$W = 97.5$, $z = -3.5$, $\mathbf{p < 0.01}$	$W = 121$, $z = -3.9$, $\mathbf{p < 0.01}$
5	$W = 49$, $z = -3.1$, $\mathbf{p < 0.01}$	$W = 100$, $z = -3.7$, $\mathbf{p < 0.01}$
6	$W = 81$, $z = -3.55$, $\mathbf{p < 0.01}$	$W = 131$, $z = -4.0$, $\mathbf{p < 0.01}$
7	$W = 100$, $z = -3.8$, $\mathbf{p < 0.01}$	$W = 100$, $z = -3.6$, $\mathbf{p < 0.01}$

Table III.11: Wilcoxon Rank Sum test results comparing computation time of the algorithms for the two communication ranges and 50 robots.

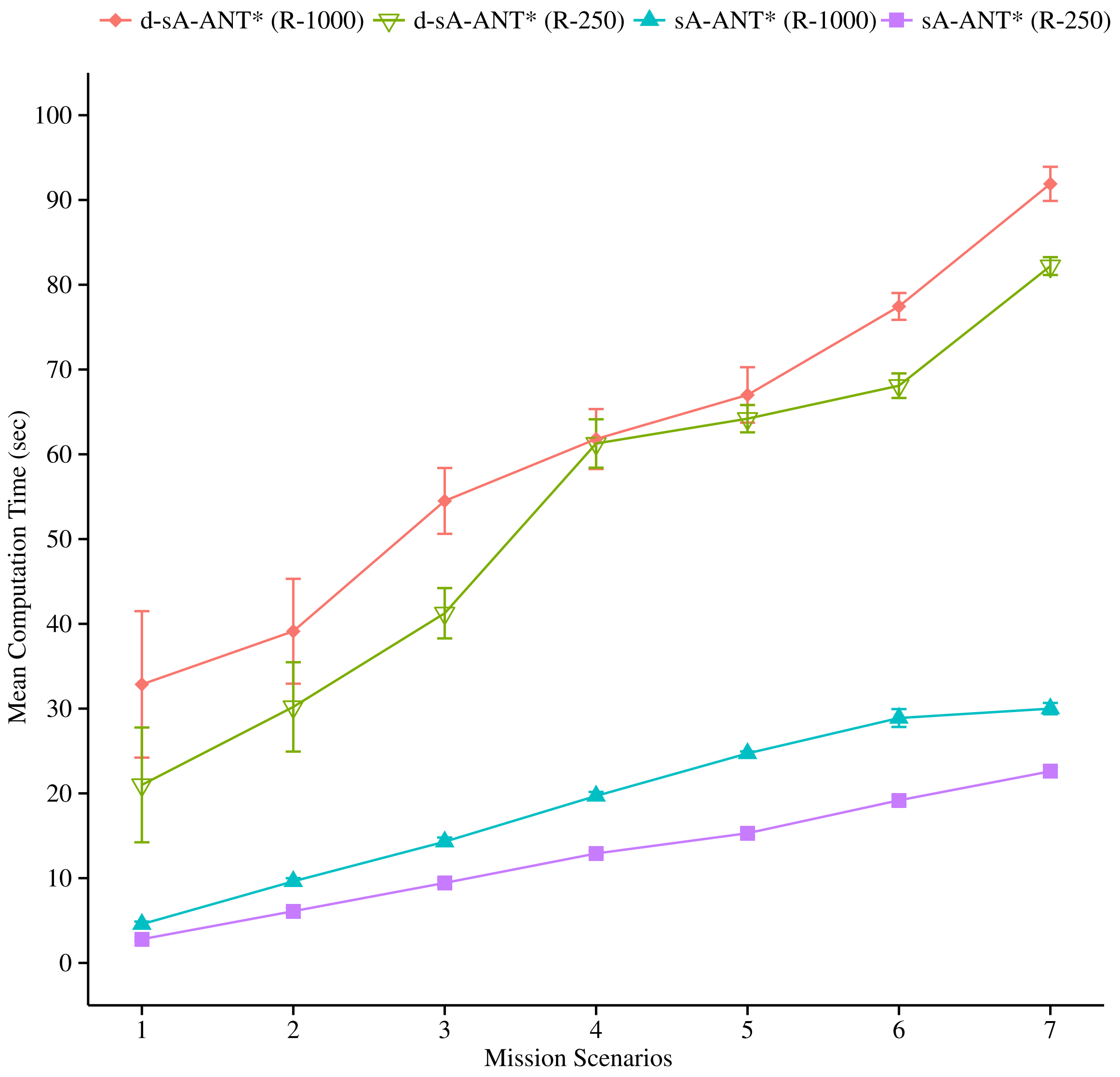


Figure III.17: The mean computation time for coalition formation for both the 250m and 1000m communication ranges and 50 robots.

III.5.2.1.2 Coalition utility

Despite being decentralized, *d-sA-ANT** generated coalitions that had virtually identical mean utilities to those computed by *sA-ANT**, irrespective of the number of robots and the communication ranges. There was no significant difference in the mean coalition utilities generated by the algorithms. Figures III.18 and III.19 provide the comparison of coalition utilities for the two communication ranges of 250m and 1000m and $n = 20$ and 50, respectively.

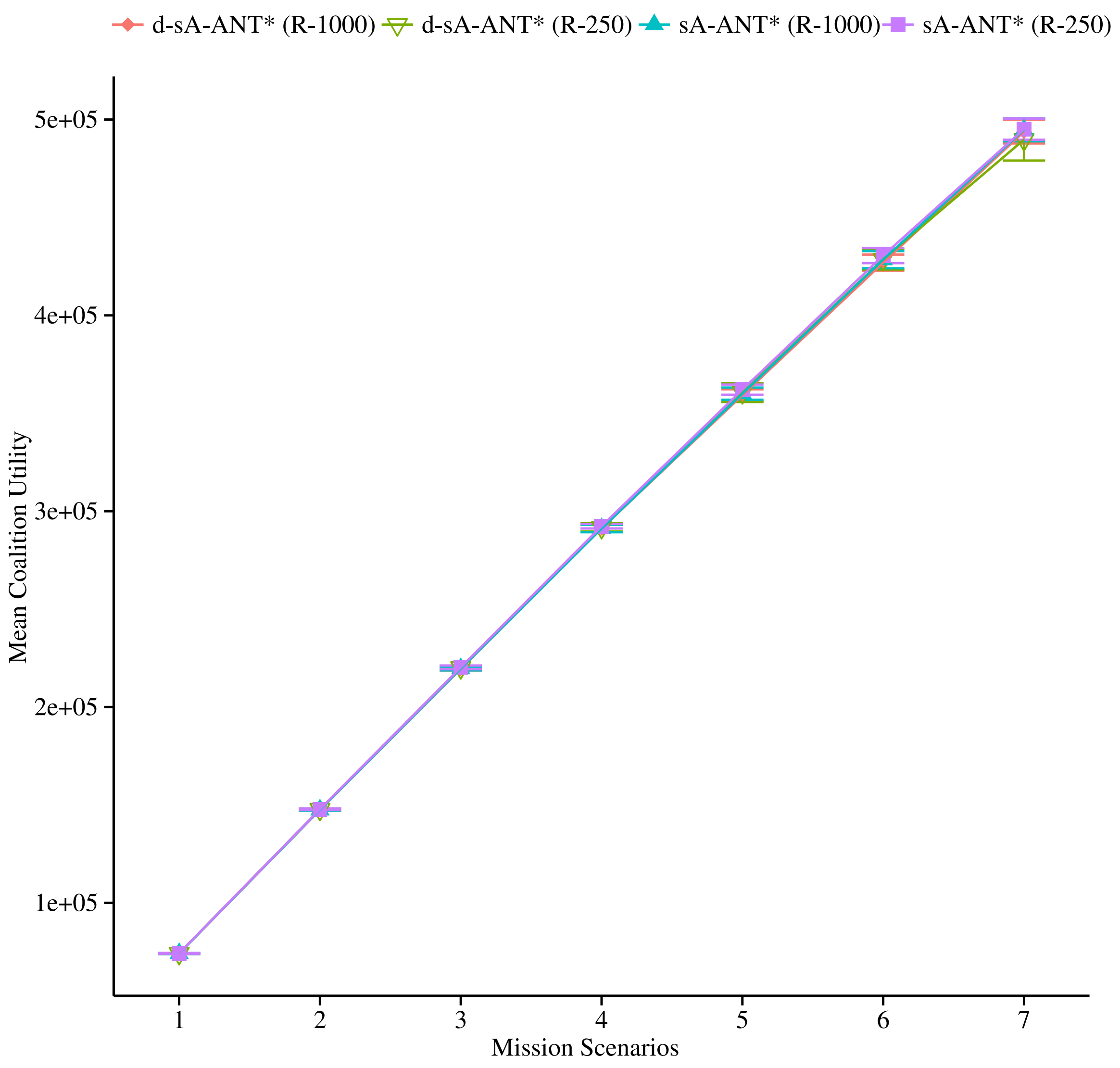


Figure III.18: The mean coalition utility for the communication ranges of 250m and 1000m ($n = 20$).

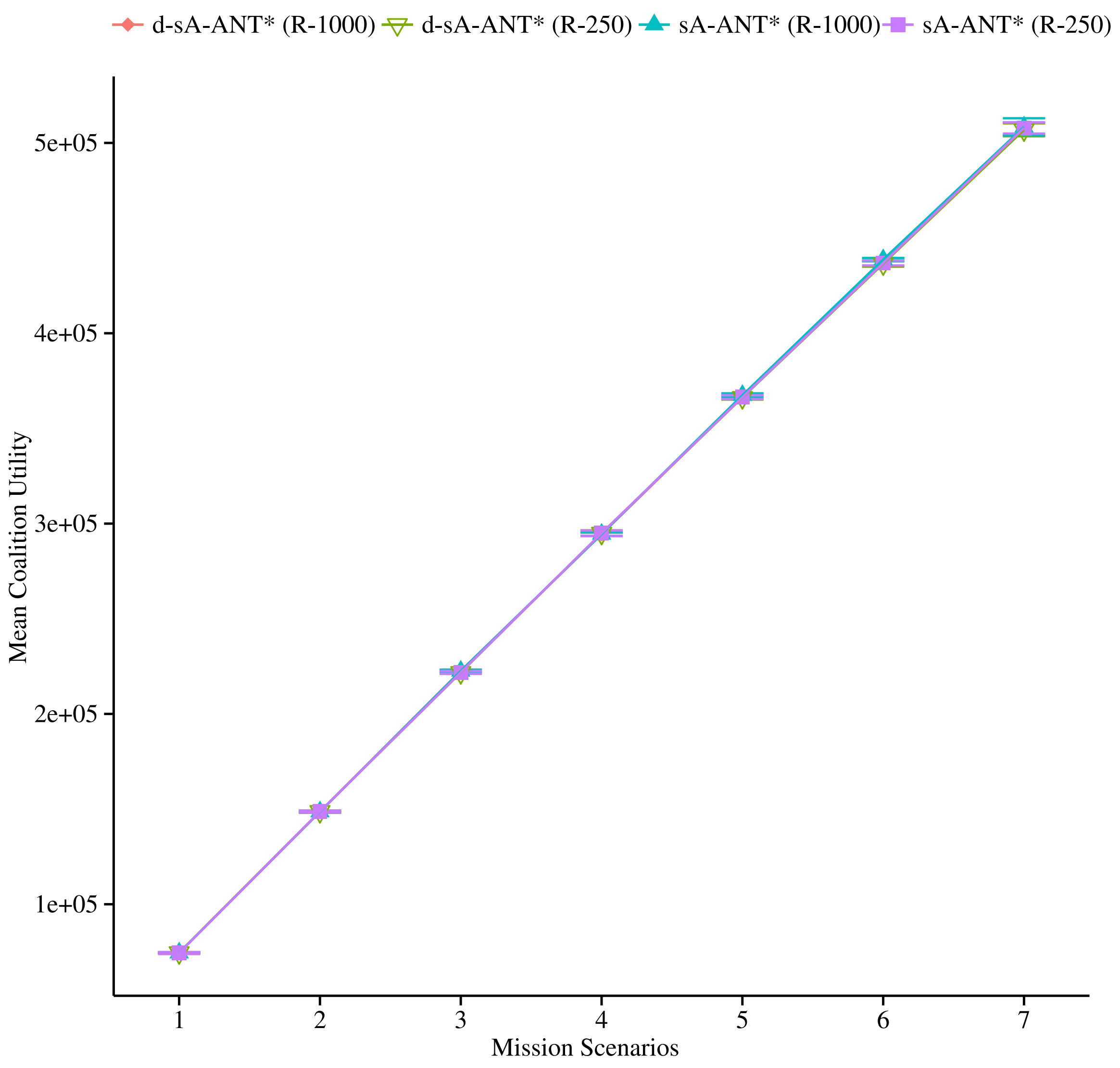


Figure III.19: The mean coalition utility for the communication ranges of 250m and 1000m ($n = 50$).

III.5.2.1.3 Travel distance

The mean traveling distance for all members of each coalition generated by *sA-ANT** and *d-sA-ANT** was virtually identical irrespective of the number of robots and the communication ranges of 250m and 1000m, as shown in Figures III.20 and III.21. A Kruskal-Wallis one-way analysis of variance determined that there was no significant main effect of algorithm on the traveling distance across the communication ranges and number of robots. It is noted that the traveling distance is lower when $n = 50$ irrespective of the communication range, which is due to a higher likelihood of finding robots that are closer to the task location.

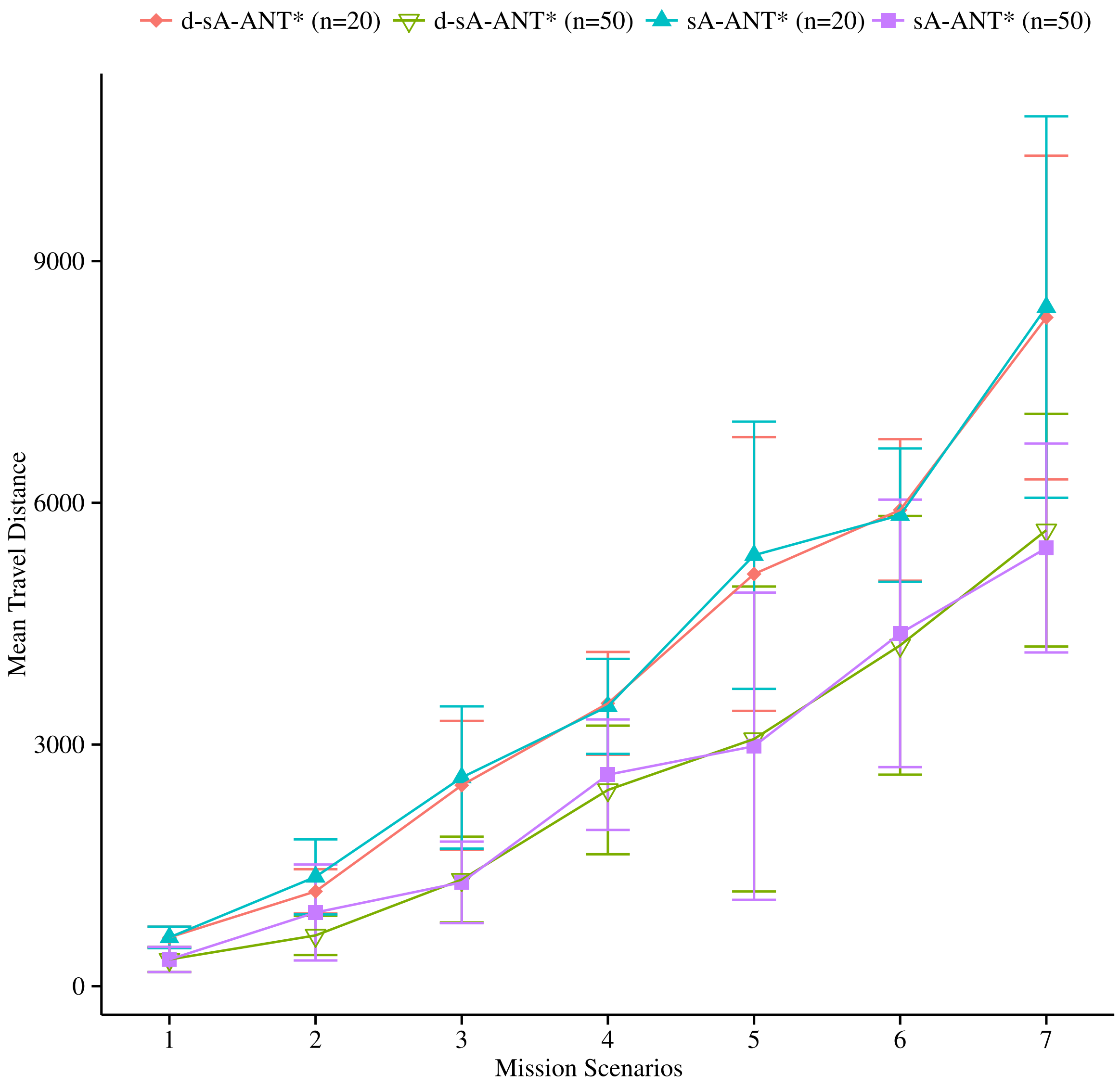


Figure III.20: The mean traveling distance across the missions for the 1000m communication range and $n = 20$ and 50.

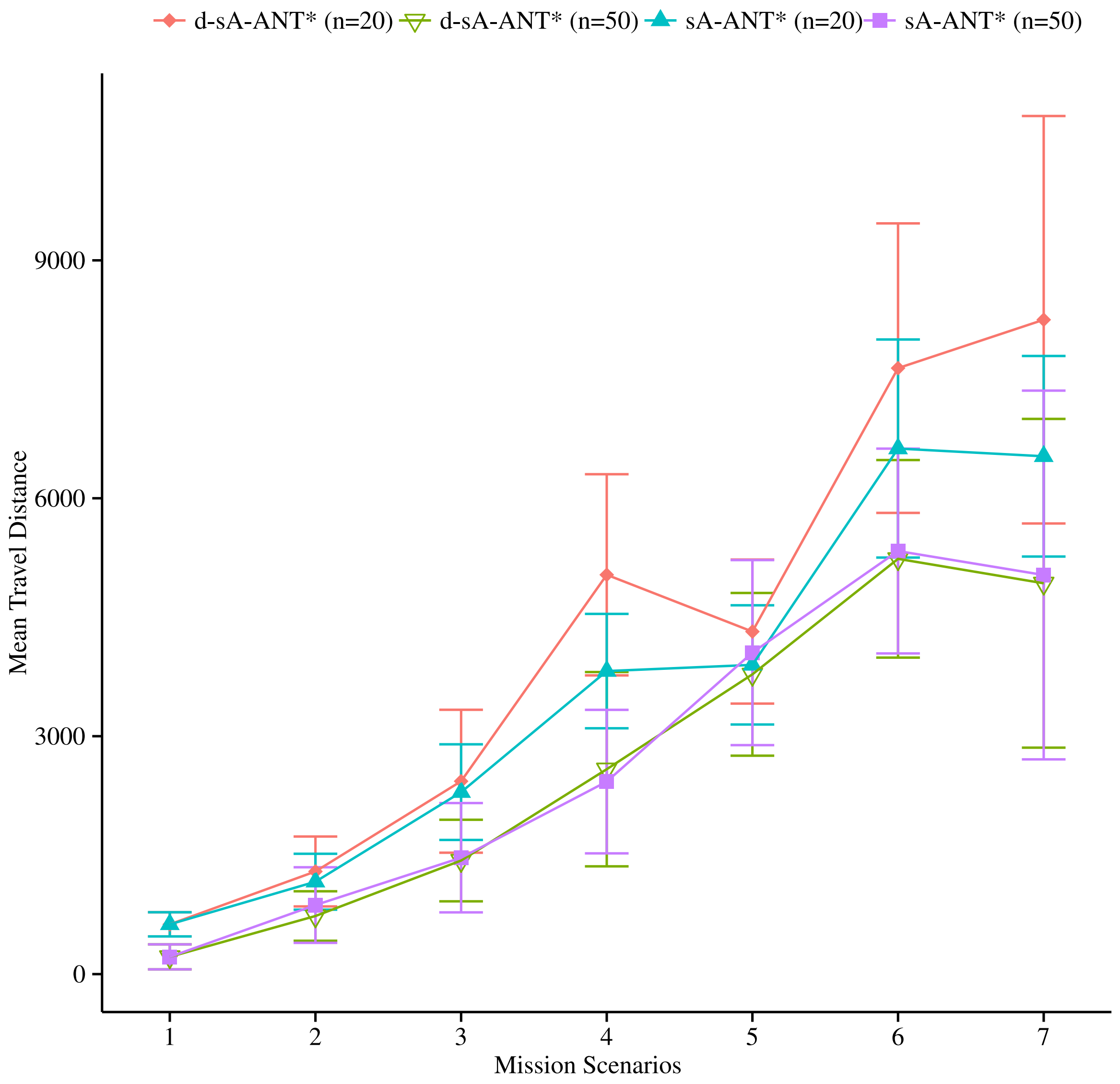


Figure III.21: The mean traveling distance across the missions for the 250m communication range and $n = 20$ and 50.

III.5.2.1.4 Coalition sizes and composition

The utilities of the coalitions generated by *sA-ANT** and *d-sA-ANT** were virtually identically, as shown in Figures III.18 and III.19. Therefore, the coalition sizes and their composition were investigated. The computed coalitions were identical both in size and composition in $\approx 72\%$ of the ten trials across the missions, communication ranges, and robot counts. However, for $\approx 16\%$, the coalition composition differed, even when the coalition sizes were identical. The coalition sizes differed for the remaining $\approx 12\%$ of trials, in which the sizes differed by one or two robots.

For example, consider a trial for the mission scenario 7, where each resource type was set to 40, the number of robots was 50, and the communication range was 1000m. *d-sA-ANT** and *sA-ANT** both generated coalitions of size 10, but the former was comprised of $\{3, 11, 14, 26, 29, 32, 36, 37, 47, 48\}$, while the latter consisted of $\{3, 11, 13, 14, 29, 32, 36, 38, 47, 48\}$ (here the number refers to robot identifier). The generated coalition utility for *d-sA-ANT** was 504552, and 505493 for *sA-ANT** with computation times of 91 seconds and 28 seconds, respectively.

Now consider another trial for the same mission, with the same number of robots and the same communication range. The coalitions generated by *d-sA-ANT** and *sA-ANT** were of the sizes 10 and 9, with the former comprised of $\{1, 6, 8, 22, 26, 30, 31, 37, 46, 47\}$, while the latter contained $\{3, 22, 26, 30, 31, 37, 38, 46, 47\}$. *d-sA-ANT**'s coalition had a utility of 506893 and a 90 second computation time, while *sA-ANT**'s utility was 507872 and required 25 seconds to compute the coalition.

III.5.3 Discussion of results

The implementation of *d-sA-ANT** leverages concurrent threads in Qt, which results in an algorithm that can be ported to real robots conveniently. The communication ranges used in the experiments are based on the ranges of real wireless routers commonly used in robot systems. Such an emulation of multiple robots using parallel threads replicates realistic real-world mission situations with multiple robots distributed across an environment, where they may be unable to communicate with all other robots or a centralized agent due to communication range constraints or communication failures, while concurrently computing coalitions. The decentralization of *sA-ANT** renders it applicable to real-world settings, when robot failures can occur at random. Since the algorithm requires an information exchange during every iteration, the knowledge pertaining to robot failures is propagated throughout the network by means of local information publishing, which provides a graceful performance degradation in such an event. However, a shortcoming of *d-sA-ANT** is its dependence on high communication bandwidth due to the heavy message passing on the existing network. Moreover, as demonstrated in the experiments, *d-sA-ANT** requires significantly higher computation time due to the information propagation overhead, as compared to *sA-ANT**, as hypothesized in H_9. The hypothesis H_{10} is supported given that the quality of the coalitions generated by *d-sA-ANT** are virtually identical to that of the centralized version, because each robot leverages a collective information, rather than just its local knowledge. The experimental results demonstrate that the total travel distance for the members of the coalitions generated by *d-sA-ANT** and *sA-ANT** were virtually identical, irrespective of the communication range and robot counts; thereby, supporting hypothesis H_{11}. It has been shown that the centralized *sA-ANT* and *sA-ANT** computed coalitions that required significantly lower travel cost when compared to *Ant-Coalition* and *Ant-Coalition-Basic*. The traveling distance is an important metric for real-world robots. The results are important, because *d-sA-ANT** is a potential coalition formation algorithm for the real-world situations.

III.6 Chapter Summary

This chapter presented two centralized variants of the hybrid ant colony optimization algorithm that incorporate the simulated annealing technique to enhance the search exploration capability of conventional ACO approaches. The algorithms introduced two novel pheromone depositing policies that prohibit the algorithms from stagnating in a local optima, a serious impediment in existing ACO algorithms. The algorithms were applied to three combinatorial optimization problems and the experimental results have shown that *sA-ANT* and *sA-ANT** outperformed existing state-of-the-art ACO algorithms when it comes to solution quality. The algorithms exhibited enhanced search exploration capability, which contributed to their improved performance. A distributed variant of *sA-ANT** has also been provided that can be applied to the multi-robot coalition formation problem. Experimental results illustrate that the computational time of *d-sA-ANT** is significantly higher than that of *sA-ANT**, because of the added information propagation overhead that is an integral part of *d-sA-ANT**. The information propagation facilitates the performance of the distributed variant, which is demonstrated by the almost identical coalition utilities created by the centralized and distributed variants of *sA-ANT**.

The presented variants of the hybrid algorithms for the coalition formation problem can be classified in accordance with Service and Adams' taxonomy as greedy algorithms that compute coalitions in real-time by requiring high communication bandwidth. The *sA-ANT*, *sA-ANT**, and *d-sA-ANT** algorithms bridge the gap in multi-agent/robot systems by permitting real-time computations for very large teams of agents/robots, without the use of conventional heuristics. However, being greedy, the algorithms do not guarantee any solution quality. Therefore, the next chapter introduces the *i-CiFHaR* coalition framework that makes intelligent decisions to determine the most appropriate subset of algorithm(s) that can be applied to a multi-criteria mission scenario.

CHAPTER IV

i-CiFHaR: Design and Implementation

IV.1 Framework Design

This chapter describes the design and implementation of the **i**ntelligent **C**oalition **F**ormation framework for **H**umans and **R**obots (*i-CiFHaR*), a smart and versatile architecture that is designed to generate robust multi-agent (robot/human) coalitions for a wide variety of real-world missions. *i-CiFHaR* incorporates a library of diverse coalition formation algorithms and leverages probabilistic reasoning in order to select the most optimized subset of algorithm(s) for generating task coalitions based on multiple mission criteria, robots' capabilities, and environmental constraints (Sen and Adams, 2014, 2013b).

i-CiFHaR is a three-tiered framework (see Figure IV.1) that includes: (1) a User Interface, (2) the Middle Level Logic Tier, and (3) a Library of coalition formation algorithms. The next sections explain the design of each tier.

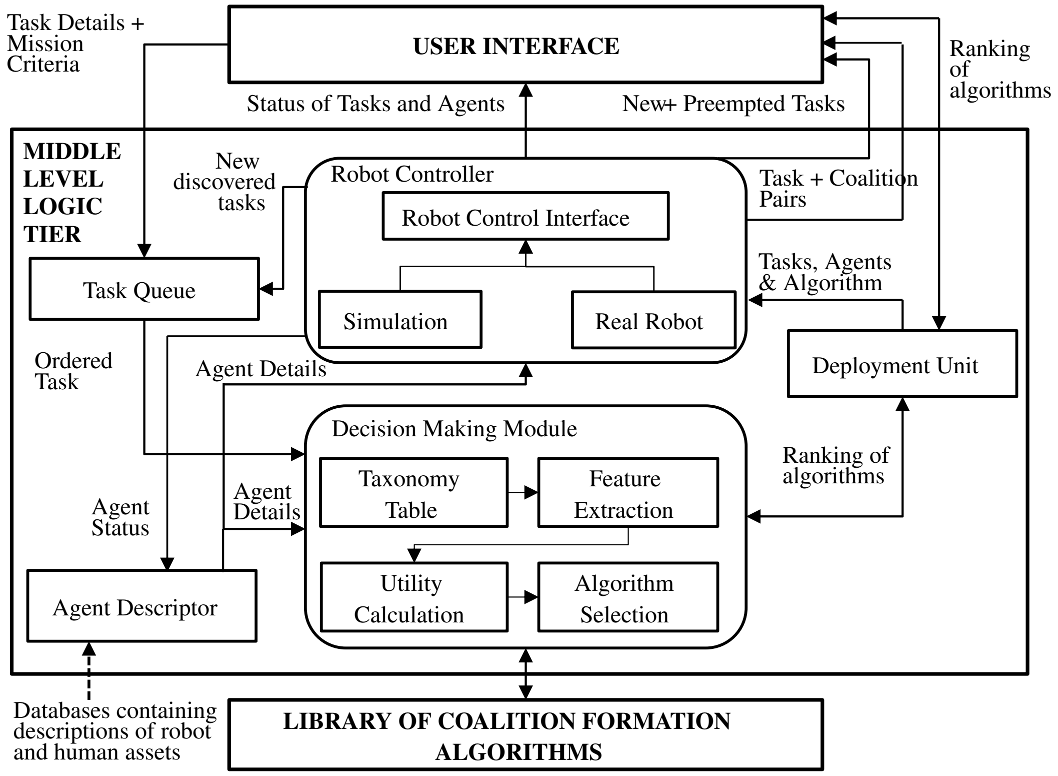


Figure IV.1: The *i-CiFHaR* architecture

IV.1.1 User Interface

The Graphical User Interface (see *User Interface* in Figure IV.1) allows a human supervisor to provide task descriptions, environment constraints, and multiple mission criteria. The interface will provide the human user with crucial information, such as: (1) ongoing task progress (e.g., executing, waiting, finished), (2) robot coalitions and task allocations, (3) individual robot status (e.g., engaged, idle, faulty), and (4) details of newly discovered or preempted tasks.

IV.1.2 Middle Level Logic Tier

The Middle Level Logic Tier accepts the mission requirements and constraints from the User Interface and the databases containing robotic and human assets' descriptions for the mission. Moreover, this tier primarily performs online probabilistic reasoning over *i-CiFHaR*'s library of diverse algorithms in order to select the most appropriate coalition formation algorithm(s) to apply.

IV.1.2.1 Task Queue

The *Task Queue* contains all the mission tasks along with their descriptions, requirements, and constraints. A *task template* (Figure IV.2) is leveraged in order to provide uniformity to task information, while storing each task in the queue.

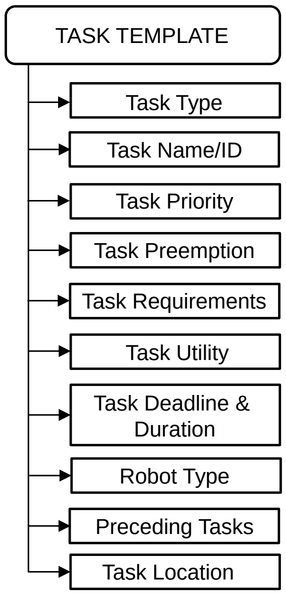


Figure IV.2: Task Template

The *task template* is a tuple and contains the following fields:

- **Task Type** - Category of a task (e.g., bomb diffuse, victim search, box pushing).

- **Task Name / ID** - Name or the Identification number of the task. The system automatically generates the task ID when a value is not provided.
- **Task Priority** - Denotes the priority or importance of the task.
- **Task Preemption** - A boolean flag to denote whether a task can be preempted or not in order to accommodate a higher priority task during a mission.
- **Task Requirements** - A vector of resources required for completing the task. Resource-based models require tasks to be described in terms of their resource requirements (e.g., camera, laser, sonar, GPS) and service-based models describes tasks by their requirements of services (e.g., box-pushing, foraging, patrolling).
- **Task Utility** - Quantifies the utility gain of a robot when a coalition completes the task.
- **Task Deadline & Duration** - Task deadline defines the latest time by which the task is to be completed and task duration is an estimated time required to complete the task uninterrupted.
- **Robot Type** - Denotes the type(s) of robot(s) required for the task (unmanned heterogeneous vehicles). *i-CiFHaR* determines the *Robot Type* based on the task requirements.
- **Preceding Tasks** - Denotes a list of tasks that need to be completed before this task can be started.
- **Task Location** - Represents either the spatial coordinates or a geographic region where a task needs to be performed. For example, a bomb diffusing task needs the location (coordinates) of the bomb, whereas a search task requires an area to be defined for the search.

Once all the tasks are entered in the *Task Queue*, the tasks require an ordering in order to satisfy the temporal task constraints. Inter-task constraints arise when: (1) execution of a particular task depends on the completion of a set of preceding task(s) (precedence constraints), and (2) tasks have temporal separations (e.g., Task T_1 can only start after 30 minutes from the the completion of a preceding task, T_2). This step is necessary because most of the coalition formation algorithms in *i-CiFHaR*'s library do not address temporal constraint satisfaction by themselves.

i-CiFHaR's temporal ordering algorithm (Algorithm 4) generates an online ordering of tasks that satisfies the temporal constraints (precedence constraints). This ordered task set will be provided to the set of most appropriate coalition formation algorithm(s) selected by *i-CiFHaR*'s decision making module, while computing the task-robot coalitions. The temporal planner algorithm acts as an important pre-processing step to abstract the temporal constraint satisfaction procedure from the coalition formation algorithms, because most

Algorithm 4 Order Planner algorithm

```
Input: $mxm$ adjacency matrix, AdjMat; Task Set, $T$ of size $m$
Output: Sorted task queue, $T_{sort}$
1: $inDegrees \leftarrow$ 1 x $m$ vector of vertices' indegrees
2: $taskStartTimes \leftarrow$ Initialize a 1 x $m$ vector of task start times with 0s
3: $taskDuration \leftarrow$ 1 x $m$ vector of task duration times
4: $T_{sort} \leftarrow$ an empty Queue
5: $zeroInDegrees \leftarrow$ Queue of all vertices with indegrees of 0
6: while $zeroInDegrees.size()$ != 0 do
7:     $v_i \leftarrow zeroInDegrees.pop\_front()$
8:     $T_{sort}.push\_back(v_i, taskStartTimes_{v_i})$
9:     for $w_j \in neighborVertices(v_i)$ do
10:        $inDegrees_{w_j} \leftarrow inDegrees_{w_j} - 1$
11:        if $inDegrees_{w_j} == 0$ then
12:            $zeroInDegrees.push\_back(w_j)$
13:            $maxValue \leftarrow max_{x \in predecessor(w_j)} \left(taskStartTimes_x + taskDuration_x + AdjMat_{x,w_j}\right)$
14:            $taskStartTimes_{w_j} \leftarrow maxValue$
15:        end if
16:    end for
17: end while
18: Return $T_{sort}$
```

of the algorithms in *i-CiFHaR*'s library assume that the mission tasks do not have any precedence constraints, an invalid assumption for real-world missions.

The temporal order planning algorithm (Algorithm 4) is based on *topological sorting*. This planner addresses the satisfaction of the temporal constraints (e.g., precedence ordering, temporal separations) of the tasks. Algorithm 4 leverages a weighted, directed, and acyclic graph, $G(V,E)$, in which each vertex $v_i \in V$ represents a task in the task set, T and a directed edge $e_{ij} \in E$ represents the dependency of task, j on the completion of a predecessor task, i. The weight, w_{ij} associated with each edge, e_{ij} represents the temporal separation between the end time of task, i and the start time of task, j. An adjacency matrix, $AdjMat$ of size $m \times m$ is created from graph, $G(V,E)$ that captures the neighbors of the i^{th} task in the i^{th} row and the predecessors of the j^{th} task in the j^{th} column (m in the number of tasks in task set, T). Algorithm 4 takes as input the adjacency matrix, $AdjMat$ and the task set, T of size m and requires an order of $O(m+y)$ computational time, where y is the number of task dependencies. The output of Algorithm 4 is the required ordering of tasks, T_{sort} that satisfies the temporal task constraints. Upon discovery of a new task during a dynamic mission (when tasks are not known a-priori), this new task is accommodated in $G(V,E)$ and processed; thereby, modifying the existing ordering in an incremental fashion.

IV.1.2.2 Agent Descriptor

The *Agent Descriptor* module contains the relevant information of all the system's agents (robots/human assets) that include: (1) robots' capabilities, (2) current robot pose, and (3) robot task status (e.g., committed to a particular coalition, uncommitted), and (4) robot state (e.g., working, faulty, standby). All the descriptions are provided by databases in XML format.

IV.1.2.3 Decision Making Module

The primary component of the *Middle Level Logic Tier* is the *Decision Making Module* that chooses the most appropriate coalition formation algorithm(s) to apply to a given situation. This module classifies the coalition formation algorithms in *i-CiFHaR*'s library along multiple dimensions, or features based on an existing taxonomy (Service and Adams, 2010). The *Decision Making Module* is comprised of the: (1) *Taxonomy*, (2) *Utility Calculation*, (3) *Feature Extraction*, and (4) *Influence Diagram*. The *taxonomy table* stores the taxonomy features and the respective domain values that facilitate classification of the coalition algorithms. The *Utility Calculation* determines the feature-value pair utility scores that are essential for creating the influence diagram's utility table (see Section IV.1.2.3.2). *Feature Extraction* determines the most important features that discriminate the algorithms, thus reducing the problem dimensionality (see Section IV.1.2.3.3). The *Influence Diagram* builds the system's influence diagram/decision network dynamically at run-time based on the extracted prominent features (see Section IV.1.2.3.4).

IV.1.2.3.1 Taxonomy Table

i-CiFHaR employs Service and Adams' existing coalition formation algorithm taxonomy that defines multiple dimensions, or features for algorithm classification (Service and Adams, 2010). Service and Adams' taxonomy encapsulates a broad set of features/dimensions for the multi-robot task allocation problem and borrows many of the dimensions from the aforementioned taxonomies (see Section II.4). The taxonomy dimensions are partitioned into four relation-based categories: (1) agent, (2) task, (3) domain, and (4) algorithm. Service and Adams classified a number of coalition formation algorithms according to the taxonomy dimensions. Table IV.1 categorizes the taxonomy features into the four categories and highlights the respective domain values.

Let F be a set that contains the N taxonomy features ($N = 18$), where each feature has its respective non-empty domain set (see Table IV.1). Let *Dom* be a collective set containing all the respective domain sets of N taxonomy features. All this information is captured by Equation IV.1, which is defined as:

$$\forall F_i \in F, \exists D_i \in Dom \mid 1 \leq i \leq N, D_i \neq \emptyset, \tag{IV.1}$$

Category	Taxonomy Features (F)	Feature Domain Values (Dom)
Agent	Agent Orientation (F_1)	{Group Rational, Self-Interested}
	Agent Type (F_2)	{Homogeneous, Heterogeneous}
	Agent Capability Model (F_3)	{Resource, Service}
	Agent Awareness (F_4)	{Aware, Partially, Unaware}
	Agent Structure (F_5)	{Social Network, Organization Hierarchy, None}
Task	Inter-Task Constraints (F_6)	{Yes, Prerequisite, No}
	Task Preemption (F_7)	{Yes, No}
	Task Requirement Model (F_8)	{Resource, Service}
	Intra-Task Constraints (F_9)	{Yes, No}
	Task Coupling (F_{10})	{Tightly, Loosely, Intermediate}
Domain	Performance Criterion (F_{11})	{Maximize Utility, Minimize Cost, Maximize Task}
	Communication Overhead (F_{12})	{High, Low}
	Task Allocation (F_{13})	{Instantaneous, Time-Extended}
	Spatial Constraints (F_{14})	{Yes, No}
	Overlapping Coalitions (F_{15})	{Yes, No}
Algorithm	Algorithm Technique (F_{16})	{Greedy, Auction-based, Approximation}
	Implementation (F_{17})	{Centralized, De-Centralized}
	Coalition Size Constraint (F_{18})	{Single, None, Fixed Upper Limit}

Table IV.1: Taxonomy Features and respective domain values (Service and Adams, 2010).

where D_i is the domain value set of feature F_i. A feature $F_i \in F$ can be instantiated with any particular value of its domain value set, D_i.

IV.1.2.3.2 Utility Calculation

Influence diagrams contain *chance nodes* representing random variables, *decision nodes*, and a single *utility node*. The *utility node* has a utility value table (degree of preference) for all possible parent node configurations. The parents of the *utility node* in *i-CiFHaR*'s influence diagram are: (1) a set of *chance nodes* representing the subset of important taxonomy features and (2) a *decision node* with its domain containing all the coalition formation algorithms. Since *i-CiFHaR*'s built-in mathematical model computes the utility table entries, the utility scores of the feature-value pairs and the algorithms need to be determined.

The utility calculation is based on *link analysis*, which has previously been used to capture the connections or associations in social networks among friends, computers in computer networks, webpages on the internet, etc. The exploration of link analysis in world wide web led to the notable applications: HITS (Kleinberg, 1999) and PageRank (Page et al., 1999) that compute composite numerical scores for web pages with the intent of measuring their relative importance. Query web pages (called *hubs*) are linked to multiple query relevant web pages (called *authorities*) in a hyperlinked environment (Kleinberg, 1999).

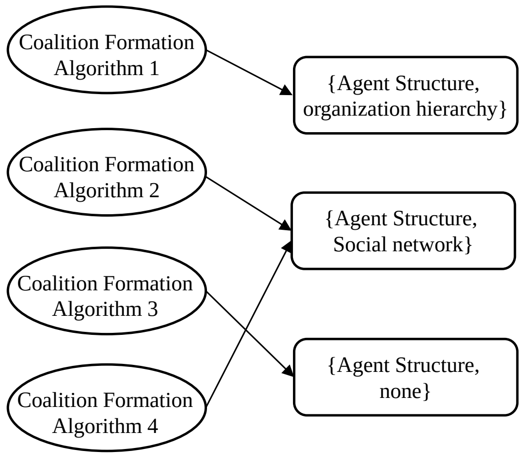

Coalition Formation Algorithm 1
Coalition Formation Algorithm 2
Coalition Formation Algorithm 3
Coalition Formation Algorithm 4
{Agent Structure, organization hierarchy}
{Agent Structure, Social network}
{Agent Structure, none}

Figure IV.3: Link graph connecting four coalition formation algorithms to three feature-value pairs of the taxonomy feature, *Agent Structure*.

Each coalition formation algorithm can be linked to a subset of related feature-value pairs that govern the algorithm's applicability (Figure IV.3); therefore, the algorithms and the feature-value pairs can be visualized as *hubs* and *authorities*, respectively. Let V be a set of size d containing all possible feature-value pairs derived from the set, F of taxonomy features and their corresponding domain sets *Dom*, such that

$$V = \{(F_x, d_i) \mid F_x \in F, d_i \in D_x, D_x \in Dom\}. \tag{IV.2}$$

The size of the feature-value pair set V is defined as $d = \sum_{x=1}^{N} |D_x|$, where $|D_x|$ represents the domain size of $F_x \in F$. A feature-value pair, $FVP_\phi \in V$ when associated with a particular taxonomy feature, $F_x \in F$ is represented as FVP_ϕ^x. Based on the associations between *i-CiFHaR*'s coalition formation algorithms and feature-value pairs, a link structure can be derived. For example, the taxonomy feature *Agent Structure (F_5)*, with domain D_5 = {*organization hierarchy*, *social network*, *none*}, and $|D_5| = 3$ results in three possible feature-value pairs: (1) {*Agent Structure, organization hierarchy*}, (2) {*Agent Structure, social network*}, and (3) {*Agent Structure, none*} (see Figure IV.3). For instance, let *i-CiFHaR* incorporate four random algorithms (Coalition Formation Algorithms 1 through 4 in Figure IV.3) and leverage only the single taxonomy feature *Agent Structure* for algorithm classification. The algorithms are connected manually to the respective feature-value pairs, thereby forming a link structure, as shown in Figure IV.3. Building on this idea, *i-CiFHaR* generates a complete link graph between the coalition formation algorithms in the library and all possible feature-value pairs in set, V in accordance with Service and Adams' taxonomy.

Let C be the set containing p coalition formation algorithms in *i-CiFHaR*'s library. The complete link graph is represented by a bipartite directed graph $G(C,V,E)$, which is constructed using C and V as the two disjoint node sets of G. A directed edge $e_{lo} \in E$ from a coalition formation algorithm, $C_l \in C$ to a feature-value pair, $FVP_o \in V$ associates C_l with feature-value pair FVP_o.

Algorithm 5 Utility Calculation algorithm

```
Input: pxd matrix, $AMat$; constant, $const$; Number of iterations, $itr$
Output: $FVPBaseScore$ of size 1x$d$
 1: $algoVector \leftarrow$ 1 x $p$ vector of 1s
 2: $FVPVector \leftarrow$ 1 x $d$ vector of 1s
 3: for i = 1 to $itr$ do
 4:     $FVPVector \leftarrow algoVector \times AMat$     // Vector-Matrix Multiplication
 5:     $FVPVector \leftarrow \frac{FVPVector}{2-Norm(FVPVector)}$
 6:     $algoVector \leftarrow FVPVector \times AMat^T$     // Vector-Matrix Multiplication
 7:     $algoVector \leftarrow \frac{algoVector}{2-Norm(algoVector)}$
 8: end for
 9: $FVPBaseScore \leftarrow const \times FVPVector$
10: Return $FVPBaseScore$
```

Motivated by the HITS algorithm (Kleinberg, 1999), the Utility Calculation algorithm (Algorithm 5) computes the base utility score of each feature-value pair. A $p \times d$ matrix, $AMat = \{a_{ij}\}$ is computed based on the complete link structure. The rows of $AMat$ represent the coalition formation algorithms, while the columns represent all possible feature-value pairs. An element, $a_{ij} \in AMat$ ($1 \leq i \leq p$, $1 \leq j \leq d$) is defined as,

$$a_{ij} = \begin{cases} 1 & \text{if } C_i \in C \text{ is associated with } FVP_j \in V \\ 0 & \text{otherwise.} \end{cases} \tag{IV.3}$$

The link structure node weights are initialized to 1 and are updated iteratively until they converge to steady-state utility values. The convergence is guaranteed because the feature-value pair utility scores constitute the principal *Eigen vector* of $AMat^T \times AMat$ (Kleinberg, 1999). During each iteration of Algorithm 5, the vectors, $algoVector$ and $FVPvector$ are normalized using an Euclidean norm (2-Norm), such that $\sum_{i=1}^{p} algoVector_i^2 = 1$, and $\sum_{j=1}^{d} FVPVector_j^2 = 1$. The constant, $const$ scales the normalized utilities. The time complexity of Algorithm 5 is $O(itr \times p \times d)$ and the utilities converge to steady-state values very quickly, with $itr \approx 20$. The generated feature-value pair base utility scores are purely a function of the link structure. Given the feature-value pair set V containing all possible feature-value pairs (FVPs), the base utility scores ($FVPBaseScore$) are calculated using the Utility Calculation algorithm (Algorithm 5) and

$$\forall FVP_\phi \in V, \exists FVPBaseScore_\phi \in FVPBaseScore, \mid 1 \leq \phi \leq d, \tag{IV.4}$$

where FVP_ϕ is the ϕ^{th} feature-value pair. These base utility scores are weighted dynamically in accordance with the mission requirements (see Section IV.1.2.3.4).

IV.1.2.3.3 Feature Extraction

i-CiFHaR leverages eighteen features; however, many features do not contribute to classifying the algorithms. For example, the feature *Agent Orientation* is assigned the same value (*Group Rational*) for all algorithms currently in the library. Feature extraction removes redundant features and reduces the problem dimensionality. *Principal Component Analysis* has been demonstrated for feature selection (Jolliffe, 1972; Song et al., 2010). *i-CiFHaR* extracts prominent features that discriminate between algorithms using a Feature Extraction algorithm similar to Song et al.'s approach for image processing (Song et al., 2010). *i-CiFHaR*'s feature selection algorithm differs in that it rejects all taxonomy features that result in zero coefficient factors across all major principal components. The procedure is formally proven in Lemma 2.

The selection algorithm leverages a $p \times N$ matrix U, where p is the number of algorithms in *i-CiFHaR*'s library and N is the number of taxonomy features. An element, $u_{ij} \in U$ is the base utility score (computed by Algorithm 5) for the specific feature-value pair, with feature j associated with algorithm i. *i-CiFHaR*'s feature selection algorithm requires $O(p \times N^2)$ time to generate the covariance matrix, *covC* of U and uses the *Singular Value Decomposition* technique to compute the *Eigen vectors*; therefore, the total time complexity of *i-CiFHaR*'s feature selection algorithm is $O((p \times N^2) + N^3)$.

Each eigenvector accounts for some variance in the original data set and is expressed as a linear combination of the N taxonomy features. The κ^{th} eigenvector, pc_κ is defined by

$$pc_\kappa = z_{\kappa 1}F_1 + z_{\kappa 2}F_2 + ... + z_{\kappa N}F_N = \sum_{i=1}^{N} z_{\kappa i}F_i = F\mathscr{Z}, \tag{IV.5}$$

where F is the row feature vector of size N with $F_i \in F$ representing the i^{th} taxonomy feature. $\mathscr{Z}$ is a matrix of size $N \times N$ containing the weight coefficients of all the N eigenvectors. The κ^{th} column of $\mathscr{Z}$ consists of all the weight coefficients of the κ^{th} eigenvector ($\kappa \in [1,N]$).

The primary statistics resulting from the κ^{th} eigenvector constitute the associated variance (λ_κ) and the weight vector ($z_{\kappa 1}, z_{\kappa 2}, ..., z_{\kappa N}$). The relative sizes of the coefficients ($z_{\kappa i}$) in the weight vector indicate the relative contributions of the corresponding feature, $F_i \in F$ in the original feature data set to the variance (λ_κ) of the eigenvector, pc_κ (Dunteman, 1989).

Lemma 2. *If a feature, $F_i \in F$ produces zero coefficient factors consistently for all the major principal components, then F_i contributes nothing towards the variance of the entire data set, σ_{data}.*

Proof. Let N be the total number of taxonomy features, then $\sigma_{data} = \sum_{i=1}^{N} \sigma_i^2$, where σ_i^2 is the variance of the i^{th} feature, F_i. Let λ_κ represents the variance of the eigenvector, pc_κ. The eigendecomposition generates eigenvector pc_κ, such that the variance of $pc_\kappa = \sum_{i=1}^{N} z_{\kappa i}F_i$ is maximized under the constraint, $\sum_{i=1}^{N} z_{\kappa i}^2 = 1$.

The weight coefficient, $z_{\kappa i}$ also represents the correlation coefficient between feature F_i and the principal component pc_κ; therefore, $z_{\kappa i} = 0$ means the angle between pc_κ and a unit vector along F_i is 90 degrees ($\because$ cosine(90°) = 0); therefore, the vectors are orthogonal. $z_{\kappa i} = 0$ means no linear dependencies exist between F_i and pc_κ, and F_i does not contribute to the variance of pc_κ, because $\lambda_\kappa = \sum_{i=1}^{N} \sum_{j=1}^{N} z_{\kappa i} z_{\kappa j} \sigma_{ij}$. Since, $\sum_{i=1}^{N} \lambda_\kappa = \sum_{i=1}^{N} \sigma_i^2 = \sigma_{data}$; therefore, F_i does not contribute to the variance of the entire data set. $\square$

		Six Principal Components/Eigen Vectors					
		EigVec1	**EigVec2**	**EigVec3**	**EigVec4**	**EigVec5**	**EigVec6**
Eigen Values		4.7	3.3	2.1	1.5	1.3	1
|Weight Coefficients|	$\mathbf{z_1}$	0	0	0	0	0	0
	$\mathbf{z_2}$	0	0	0	0	0	0
	$\mathbf{z_3}$	0.14	0.09	0.24	0.03	0.05	0.16
	$\mathbf{z_4}$	0	0	0	0	0	0
	$\mathbf{z_5}$	0.14	0.57	0.45	0.07	0.23	0.16
	$\mathbf{z_6}$	0.21	0.16	0.16	0.14	0.03	0.19
	$\mathbf{z_7}$	0.16	0.05	0.2	0.03	0.72	0.16
	$\mathbf{z_8}$	0.14	0.09	0.24	0.03	0.05	0.16
	$\mathbf{z_9}$	0.38	0.28	0	0.18	0.11	0.02
	$\mathbf{z_{10}}$*	0	0	0	0	0	0
	$\mathbf{z_{11}}$	0.42	0.07	0.03	0.33	0	0.52
	$\mathbf{z_{12}}$	0.03	0.39	0.24	0.21	0.22	0.23
	$\mathbf{z_{13}}$	0.47	0.02	0.3	0.19	0.01	0.43
	$\mathbf{z_{14}}$	0.4	0.28	0.07	0.32	0.26	0.44
	$\mathbf{z_{15}}$	0.17	0.04	0.33	0.73	0.18	0.37
	$\mathbf{z_{16}}$	0.33	0.31	0.37	0.01	0.1	0.11
	$\mathbf{z_{17}}$	0	0.47	0.41	0.03	0.51	0.09
	$\mathbf{z_{18}}$	0.17	0.07	0.25	0.37	0	0.09

**Weight factors are very small (in order of* 10^{-19}*), and thus rounded to 0*

Table IV.2: Weight Coefficients of the first six principal components or Eigen Vectors (EigVec$x$).

Eighteen *Eigen Vectors* or principal components are computed by *i-CiFHaR*'s feature extraction algorithm. Six components (the EigVec$x$ in Table IV.2, where $1 \leq x \leq 6$) account for approximately 94% of the total variance in the original data set. These six principal components and their associated variances (Eigen Values) are enumerated in Table IV.2. Each principal component comprises a weight vector containing eighteen weight coefficients corresponding to the taxonomy features. The weight coefficients of the taxonomy features F_1, F_2, F_4, and F_{10} are zero for all of the major principal components and do not contribute significantly to the classification (shaded gray in Table IV.2). The remaining features become *chance nodes* in the influence diagram.

IV.1.2.3.4 Influence Diagram Construction

Once the Feature Extraction algorithm identifies the most prominent features, the influence diagram is built dynamically at run-time. An influence diagram augments a Bayesian network by introducing decision variables and a utility function that characterizes the decision maker's (here *i-CiFHaR*) preferences. *i-CiFHaR* solves the decision problem by determining the optimal strategy that maximizes the expected utility score for the framework.

The prominent taxonomy features are the most influential and uncertain criteria that can be leveraged to discriminate between the coalition formation algorithms. The influence diagram's *decision node* contains the decision alternatives that are mutually exclusive, finite, and exhaustive. Since *i-CiFHaR* seeks to select the most optimal algorithm(s) to apply at a specific time, the *decision node*'s domain consists of all the coalition formation algorithms in its library. The random *chance nodes* and the *decision node* become the parents of the single *utility node*, which holds a utility table for all possible configurations of the parent nodes. During a real-world mission scenario, the incomplete information regarding the situation is captured in terms of probability values for each of the *chance nodes*. Given that all the *chance nodes* and the *decision node* are parents of the the *utility node*; the utility function represents all the taxonomy features and the algorithm scores. The utility table values are usually obtained by consulting domain experts or through intuition and preferences of the system designer (Yu and Terzopoulos, 2007); however, *i-CiFHaR* calculates the utility table entries automatically. The number of entries is exponential in size to the number of parents to the *utility node*. The influence diagram's utility table size (U_{size}) is given by:

$$U_{size} = p \times \prod_{x=1}^{N_{extr}} |D_x|, D_x \in Dom, F_x \in F_{Prom}, \tag{IV.6}$$

where N_{extr} is the number of extracted taxonomy features; D_x is the domain set of feature F_x; and F_{Prom} is the Prominent Feature Set containing all prominent taxonomy features, with $|F_{Prom}| = N_{extr}$.

The exponential utility table size prohibits calculating the entries based on designer preferences or intuitions. Two approaches are implemented. First, the most prominent features are used to construct the influence diagram, reducing the problem dimension. Second, a mathematical model automatically generates the utility table entries by leveraging the base utility scores of the feature-value pairs, as computed by the *Utility Calculation* module. However, a direct use of the base scores for the utility table entry calculation has a major drawback. The link-analysis algorithm computes the feature-value pair utility scores based on the link structure; thus, the more in-links to a feature-value pair, the higher its score. The feature-value pairs with low endorsements, i.e., a feature-value pair not associated with many algorithms receives a very low utility score. When feature-value pairs with low utility scores are required for a mission, the corresponding coalition algorithm is often not selected because *i-CiFHaR* seeks to maximize the system's expected utility score. Moreover, a persistent utility function makes the framework susceptible to inconsistencies in decision making. Addressing the stated drawbacks, *i-CiFHaR*'s presented implementation employs an adaptive utility function. Under the neoclassical approach, only the probabilities of the *chance nodes* vary with new information; however, no such provision exists for modifying the decision maker's preferences. Therefore, adaptive utility functions have been advocated (Cohen and Axelrod, 1984; Cyert and Degroot, 1979; Nielsen and Jensen, 2004) and it has been shown that the expected utility hypothesis still holds (Grne-Yanoff and Hansson, 2009).

An adaptive utility function through *dynamic* scoring of the feature-value pairs addresses the aforementioned drawbacks, where the base utility scores (computed by Equation IV.4) are dynamically weighted in accordance with the mission requirements. This approach renders *i-CiFHaR* more responsive to real-world mission scenarios. Each mission is described in terms of the feature-value pair assignments (FVP) for the prominent features. The mission uncertainties are captured using probability values (Pr) for each feature-value pair assignment. The mission dependent utility score ($FVPMissionScore^i_\varsigma$) of the ς^{th} feature-value pair ($FVP_\varsigma \in V$) that corresponds to the feature $F_i \in F_{Prom}$ is defined by:

$$\forall FVP_\varsigma \in V, FVPMissionScore^i_\varsigma = \exp^{\{\alpha \times \sqrt[\beta]{|Pr_\varsigma - avg_i|}\}} \times FVPBaseScore^i_\varsigma, \tag{IV.7}$$

where $\alpha = sgn(Pr_\varsigma - avg_i)$ is the signum function, $\beta = |D_i|$ is the domain size of the feature $F_i \in F_{Prom}$, and $avg_i = 1/\beta$ is the equal likelihood of F_i being assigned to any one of its domain values. The probability assigned to the feature-value pair is denoted by Pr_ς. The motivation for leveraging the aforementioned scaling approach stems from the fact that the mutual exclusion property of each taxonomy feature's domain values permits the calculation of the deviation for a particular feature-value pair $FVP_\varsigma \in V$ from avg_i of the corresponding feature, given the mission criteria. Based on the n^{th}-root exponential function (Equation IV.7), the weighting factor is greater than 1 when the deviation is positive and when the deviation is negative, the base feature-value pair score is weighted by a factor < 1. Figure IV.4 illustrates the effectiveness of *i-CiFHaR*'s n^{th}-root exponential weighting function, when compared to a conventional exponential function ($exp^{\{Pr_\varsigma - avg_i\}}$) and a base linear function. The figure demonstrates that for the two exemplary features with domain sizes two and three, *i-CiFHaR*'s weighting function generates a larger variation in the weight factors of the domain values, given their probabilities.

Consider a single taxonomy feature, F_i=*Task Preemption* with the domain set $D_i = \{Yes, No\}$. The mean assignment value (avg_i), assuming equal probability for assigning F_i to one of its two domain values, is $avg_i = 0.5$. For example, let a mission require the feature-value pair $\{TaskPreemption, Yes\}$. Assume a high

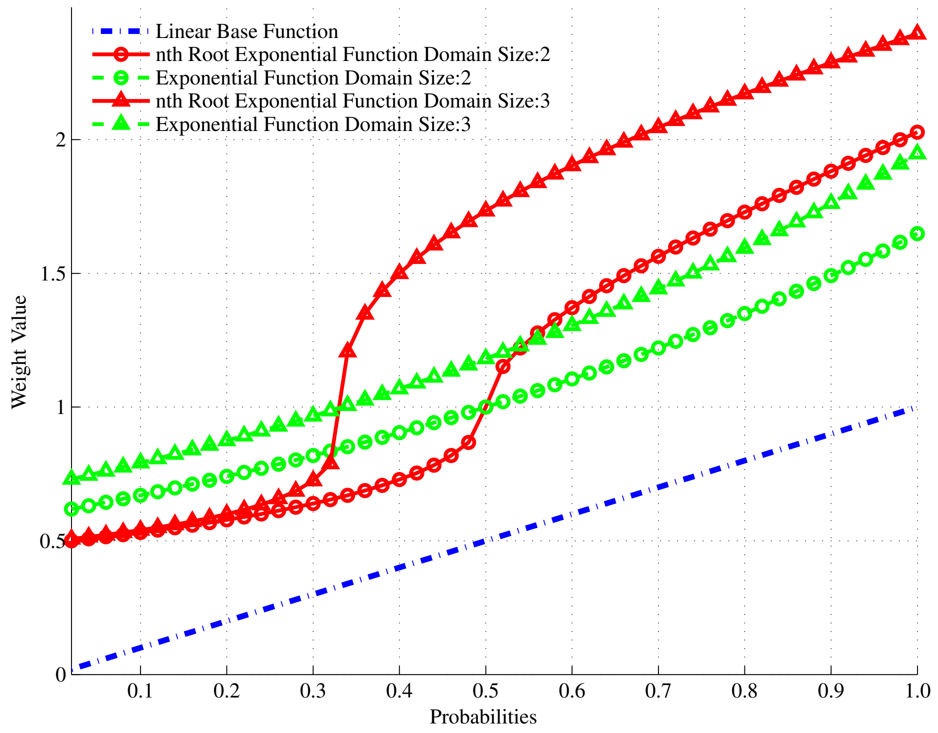


Figure IV.4: *i-CiFHaR*'s Dynamic Weighting Function. For illustration purposes, two pairs of curves are highlighted: one for a feature with domain size = 2 (shown with *circle* markers), and the other for a feature with domain size = 3 (shown with *triangle* markers). *i-CiFHaR*'s weighting function (in *red*) is compared to an exponential function (in *green*) and a base line linear function (in *blue*) for each of the features. It is noted that for the random features with domain sizes 2 and 3, the corresponding curves intersect at weight value = 1 for probabilities 0.5 and 0.33, respectively. Under such circumstances, the base utility scores of the feature-value pairs are not scaled, because each of the possible values in the features' domains has equal likelihood of being selected.

confidence in this assignment, then the assignment probability value, $Pr_{\varsigma} = 0.8$. The base utility score of $\{TaskPreemption, Yes\}$ is very low, as computed by the Utility Calculation algorithm (Algorithm 5), because only two coalition formation algorithms permit task preemption. The low utility score increases the likelihood of false positives; however, if the mission requirements are considered, the low utility score is weighted by the factor $exp^{\{1\times \sqrt[2]{|0.8-0.5|}\}} = 1.73$. Consider another mission scenario in which task preemption is not required. Let the probability value Pr_{ς} be 0.2. The mission dependent utility score is weighted by $exp^{\{-1\times \sqrt[2]{|0.2-0.5|}\}} = 0.57$.

Once the mission dependent feature-value pair utility scores are generated, each coalition formation algorithm is assigned a utility score. The intermediate utility score is derived using only the associated feature-value pairs that belong to the prominent taxonomy features, as calculated by:

$$\forall C_l \in C, algoScore_l = \sum_{\varsigma} a_{l\varsigma} \times FVPMissionScore^i_{\varsigma} \\ \mid 1 \leq l \leq p, F_i \in F_{Prom}, FVP^i_{\varsigma} \in V, \tag{IV.8}$$

where $a_{l\varsigma} \in AMat$ and p is the number of coalition formation algorithms in *i-CiFHaR*'s library. Equation IV.8 leverages the adjacency matrix, *AMat* that captures the associations between the algorithm and the feature-value pairs (Equation IV.3). The intermediate utility scores depend on the complete link structure used by Algorithm 5 to calculate the base utility scores. Once the algorithms' intermediate utility scores are calculated, the final mission dependent utility scores are obtained by normalizing the intermediate scores:

$$\forall C_l \in C, algoMissionScore_l = const \times \frac{algoScore_l}{2 - Norm(algoScore)} \mid 1 \leq l \leq p, \tag{IV.9}$$

where $2 - Norm(algoScore) = \sqrt{\sum_{i=1}^{p} algoScore_i^2}$ is the Euclidean norm leveraged for the normalization (similar to that in Algorithm 5). The constant, *const* scales the normalized mission dependent utility scores and is set to the same value as that in Algorithm 5. The dynamic utility scores are mission specific, which result in *i-CiFHaR* being more adaptable to a wide-range of real-world missions.

Based on the mission dependent utility scores of the coalition algorithms and the feature-value pairs, *i-CiFHaR*'s adaptive mathematical model generates the utility table entries automatically for the influence diagram, as defined by:

$$\forall St_{\upsilon} \in W, U(St_{\upsilon}|Act_i) = algoMissionScore_i \times \sum_{j=1}^{N_{extr}} a_{i\varsigma} \times FVPMissionScore^j_{\varsigma}, \tag{IV.10}$$

where $algoMissionScore_i$ is the mission dependent utility score of the i^{th} coalition formation algorithm (defined by Equation IV.9) and $FVPMissionScore^j_{\varsigma}$ is the ς^{th} feature-value pair associated with the j^{th} prominent feature, $F_j \in F_{Prom}$ (defined by Equation IV.7). Each state of the world, $St_{\upsilon} \in W$ is defined in terms of feature-value pairs of the prominent taxonomy features. The mathematical model (Equation IV.10) requires a $p \times d$ adjacency matrix, $AMat = a_{i\varsigma}$, where $a_{i\varsigma} \in AMat$ is defined by Equation IV.3. The number of prominent taxonomy features extracted by *i-CiFHaR*'s *Feature Extraction* module is N_{extr}. *i-CiFHaR*'s utility function, $U(St_{\upsilon}|Act_i)$ maps from every state St_{υ} of the hypothetical world W to a value, when an action Act_i is taken. Act_i indicates that the *decision node* chooses the i^{th} coalition formation algorithm, $C_i \in C$.

The *i-CiFHaR* framework behaves as a self-interested decision making agent and considers all the possible hypothetical states of the world as outcomes (χ) of a lottery. According to microeconomic utility theory, such a rational agent models its interest by quantifying each possible outcome using a utility/reward value that captures the agent's preference. A utility function, $U : \chi \to \mathbb{R}$ maps from the world states to real numbers. Given a particular coalition formation algorithm, *i-CiFHaR* has a preference for each of the possible world states; therefore, $\forall (St_i, St_j) \in W, \exists St_i \succ St_j$, or $St_j \succ St_i$ or $St_i \sim St_j$, where $\succ$ denotes strict preference, while $\sim$ denotes indifference. This completeness of *i-CiFHaR*'s preferences over all possible world states represents that *i-CiFHaR* either strictly prefers one state to the other, or is indifferent between the two, given a particular coalition formation algorithm. Moreover, the preference is transitive, i.e., if $St_i \succ St_j$ and $St_j \succ St_k$, then $St_i \succ St_k$. Since all the possible world states represent outcomes of a lottery, there exists a probability distribution over the states expressed as $[p_1 : St_1, p_2 : St_2, ..., p_k : St_k]$, where $\sum_k p_k = 1$. The von Neumann-Morgenstern expected utility model (Neumann and Morgenstern, 1947) states that if an agent's preference relation satisfies completeness, monotonicity, and transitivity, then there exists a utility function that satisfies: (1) $U(St_1) > U(St_2)$, iff $St_1 \succ St_2$, and (2) $EU([p_1 : St_1, p_2 : St_2, ..., p_k : St_k]) = \sum_k p_k \times U(St_k)$, where $EU(\bullet)$ denotes the *Expected Utility* of a given lottery/gamble.

i-CiFHaR's utility function (Equation IV.10) maps every possible world state to a preference utility score $\in \mathbb{R}_+$, given the algorithms. Seeking to solve the multicriteria decision problem at hand, each state $St_\upsilon \in W$ is expressed in terms of feature-value pairs of the taxonomy features; therefore, St_υ is a conjunction of $F_i \in F_{Prom}$. Additionally, the probabilities associated with each state, St_υ are translated into the joint probability distribution over all the prominent taxonomy features.

Theorem 1. *i-CiFHaR's utility function represents its preferences over all possible choices.*

Proof. Equation IV.10 defines $U(St_\upsilon | Act_i)$, which calculates the utility value of every possible world state $St_\upsilon \in W$, given an algorithm choice, Act_i. The base utility score ($FVPBaseScore^j_\varsigma$ as computed by Algorithm 5) of each feature-value pair is *const* (See Algorithm 5) times the weight coefficient of the corresponding pair in the principal *Eigenvector* of $AMat^T \times AMat$, where $AMat$ is the adjacency matrix representing the connections in the link graph. Each algorithm's utility score is derived from Equations IV.8 and IV.9 and is given by $algoMissionScore_i = k \times \sum_\varsigma a_{i\varsigma} \times FVPMissionScore^j_\varsigma$, where $k = \frac{const}{2 - Norm(algoScore)}$ is a constant, and j corresponds to $F_j \in F_{Prom}$. $FVPBaseScore^j_\varsigma$ is probabilistically scaled to achieve $FVPMissionScore^j_\varsigma$; therefore, the algorithm utility score, $algoMissionScore_i$ is constant ($\mathscr{K}_i$) for a given mission instance. *i-CiFHaR*'s utility function can be re-written as $U(St_\upsilon | Act_i) = \mathscr{K}_i \times \sum_{j=1}^{N_{extr}} a_{i\varsigma} \times FVPMissionScore^j_\varsigma$, where $a_{i\varsigma}$ is 1 if algorithm Act_i is connected to ς^{th} feature-value pair in the link graph, 0 otherwise. The world states in W are expressed by every possible configuration of the prominent taxonomy features. A state, St_α receives a higher utility than that of St_β, only if the former contains the conjunction of feature-value pairs that are connected to Act_i in the link graph. A valid utility function depends on the ordinality, rather than cardinality of the states; therefore, *i-CiFHaR*'s preferences over possible world states are correctly modeled by $U(St_\upsilon|Act_i)$, which generates $U(St_\alpha) > U(St_\beta)$, if and only if $St_\alpha \succ St_\beta$ for algorithm Act_i.

□

Once the utility table entries are generated (Equation IV.10), *i-CiFHaR* leverages the influence diagram to optimize the algorithm selection process by maximizing its expected utility score. The expected utility score ($EU(Act_i)$) for each algorithm decision (Act_i) is:

$$EU(Act_i) = \sum_{\upsilon=1}^{|states|} Pr(St_\upsilon|Act_i) \times U(St_\upsilon|Act_i), \tag{IV.11}$$

where $states$ is the subset of all possible hypothetical world states in the hypothetical world W, where the action Act_i can be selected; $U(St_\upsilon|Act_i)$ is the utility of the particular world state St_υ, which is derived from the network's utility table (Equation IV.10). *i-CiFHaR* selects the most appropriate algorithm (Act^*) to apply to a given mission scenario that maximizes the expected utility score, i.e.,

$$Act^* = \underset{Act_i \in C}{\operatorname{argmax}} EU(Act_i), \tag{IV.12}$$

where C is the set containing p coalition formation algorithms.

The objective is to provide decision support to a human mission supervisor; therefore, *i-CiFHaR* may select a subset of algorithm(s) most applicable to the mission scenario when a single algorithm does not meet all mission requirements. This set of algorithm(s) includes Act^* and all other algorithms with expected utility scores greater or equal to the threshold (EU^*), defined by:

$$EU^* = \mu \times \max_{Act_i \in C} EU(Act_i), \tag{IV.13}$$

where μ is the desired fraction of the maximum expected utility score that is required by the mission supervisor. A supervisor requiring 100% performance will obtain the best algorithm that has the maximum expected utility score, while a 90% performance will select all algorithms that have their expected utility scores greater than $EU^* = 0.9 \times \max_{Act_i \in C} EU(Act_i)$.

IV.1.2.3.5 Learning Structures among Algorithms

i-CiFHaR's influence diagram considers all the algorithms in the library during the decision making process, many of which may not be applicable to a given mission scenario. Therefore, the computational time for the online reasoning is directly proportional to the exponential utility table size of the influence diagram, which affects adversely the scalability of *i-CiFHaR* with an increase in the number of algorithms and taxonomy features. Therefore, *i-CiFHaR* mines patterns in its suite of algorithms by employing conceptual clustering, an unsupervised machine learning approach in order to extract the most suitable cluster of algorithms for application analysis during a specific mission; thereby, accomplishing better scalability and reduced computational time.

i-CiFHaR leverages COBWEB, a conceptual clustering algorithm (Fisher, 1987) for identifying clusters of similar algorithms in the library. This improved approach uses the probabilistic metric, *category utility* (Gluck, 1985) to identify the most suitable cluster of algorithms. Based on the selected cluster, the influence diagram optimizes the ranking of the algorithms in the chosen cluster by maximizing the expected utility scores. The application of conventional clustering approaches (Dempster et al., 1977; MacQueen, 1967) that employ distance-based objective functions in order to discern the partitioning of *i-CiFHaR*'s coalition formation algorithms is inappropriate, because none of the taxonomy attributes that *i-CiFHaR* leverages have numerical domain sets. Although the Generality-based Concept Formation (GCF) (Talavera and Béjar, 2001) performed similarly to COBWEB, its user dependency for hierarchy levels and generality degree necessitates a human in the loop, which is undesirable for real-world missions. Thus, the original COBWEB conceptual clustering algorithm is incorporated into *i-CiFHaR* for mining patterns among the library's algorithms that are described by nominal attributes.

Each of *i-CiFHaR*'s algorithms, C_x representing a data point in the context of clustering, is characterized by its respective vector, $CV_x = \{(F_i, d_{ij})\}$ of attribute-value pairs. Although the algorithms are associated with certain attribute-value pairs, a real-world mission scenario is highly dynamic with uncertain or missing information. Such uncertain missions are represented using a general model for nominal or categorical data with uncertainty. Under this uncertainty nominal model, each mission situation is represented by a vector of *uncertain categorical attributes* (UCAs), each of which is assigned to one of the attribute's nominal domain values with some probability that signifies the event's likelihood. Each UCA is represented by a probability distribution over the attribute's domain set. Let an attribute, $F_i \in F$ be assigned a particular value, d_{ij} from F_i's domain set, D_i with some probability p_{ij}. Assuming the cardinality of F_i's domain set, $D_i = |D_i|$, the attribute's probability distribution over all the domain values is governed by $\sum_{j=1}^{|D_i|} p_{ij} = 1$. Therefore, each mission situation, MS_y is represented by a vector, MV_y of uncertain nominal attribute-value pairs, $\{(F_i, d_{ij}, p_{ij})\}$.

i-CiFHaR incorporating COBWEB's conceptual clustering algorithm (Fisher, 1987) partitions the library's coalition formation algorithms into clusters. COBWEB incrementally builds a hierarchical classification tree of concept nodes without a predefined number of clusters. COBWEB starts with an empty root node and each algorithm, expressed as a vector of nominal attribute-value pairs is added to an incremental classification tree, one at a time. COBWEB performs a hill-climbing search through the classification space, which is governed by the *category utility* heuristic (Fisher, 1987; Gluck, 1985). The *category utility* metric is a tradeoff between intra-class similarity and inter-class dissimilarity. Given a cluster, C_k, the *category utility* metric is defined as:

$$CU(C_k) = P(C_k)[\sum_i \sum_j P(F_i = d_{ij}|C_k)^2 - \sum_i \sum_j P(F_i = d_{ij})^2], \tag{IV.14}$$

where $P(\bullet)$ defines the probability, $F_i \in F$ denotes the taxonomy attribute, and $F_i = V_{ij}$ represents an attribute-value pair, when F_i is assigned to the j^{th} domain value, $d_{ij} \in D_i$. $P(F_i = d_{ij}|C_k)^2$ defines the intra-class similarity and represents the expected number of attribute-value pairs correctly guessed, given a particular class (Fisher, 1987). $P(F_i = d_{ij})^2$ represents the expected number of attribute-value pairs guessed when no classification is provided.

COBWEB (Fisher, 1987) creates a classification tree, where the root node represents the concept containing all data observations (*i-CiFHaR*'s coalition formation algorithms), while the leaf nodes represent singleton concepts, each containing an individual observation. Each node either contains singleton concepts, or subsumes other sub-concepts. Additionally, each node holds the attribute-value counts of all the objects that it contains; therefore, representing a probability concept label. COBWEB incrementally absorbs a new object into the existing hierarchy, while employing four operators recursively in order to classify the object into the best matching concept. Given a node, the *addition* operator adds the new object to one of the node's children and computes the CU score for each case, with the objective of identifying the best two concept clusters that can house the new object. The *create a new class* operator generates a new singleton concept containing only the new observation and adds this concept to the given node. COBWEB attempts to counter the ill-effects of initially skewed data by introducing two operators. The *merge* operator combines the two best hosts into a new combined concept, which is accepted as a better partition, if and only if the CU score is higher than the previously generated clusters. The *split* operator decomposes the best concept into multiple concept clusters.

The COBWEB's search is heuristic; thus, the generated classification tree varies across multiple trials of the algorithm depending on the ordering of the data set. *i-CiFHaR* mitigates the influence of an initially skewed data set by randomly selecting an initial algorithm seed, followed by an iterative selection of a different algorithm data point that maximizes the Manhattan distance between it and the previous n seeds. *i-CiFHaR* incrementally derives a classification tree for each of thirty trials, and on each instance, calculates the partition utility score, $PU = \frac{1}{m}\sum_{k=0}^{m-1} CU(C_k)$ of the partition structure containing m clusters at first level of the tree (Level-0 denotes the root node). The tree with the maximum PU score is deemed the best partitioning of the coalition algorithms, given the objective function. Once *i-CiFHaR* identifies the optimal classification tree with the maximum *PU* score, uncertain real-world missions, described in terms of a vector of uncertain attribute-value pairs are classified according to this best hierarchical tree.

The original COBWEB conceptual clustering technique is used to pre-compute the optimal clustering hierarchy of the coalition formation algorithms in *i-CiFHaR*'s library, because each of the algorithms is described in terms of the taxonomy attributes containing nominal domain values with complete certainty. This offline processing of the algorithms' hierarchical partitioning is justified, because the system library will not change during the real-world applications. However, during the online classification of the uncertain mission scenarios, all the mission attribute-value pairs are assigned likelihood probabilities to simulate uncertainties in the real-world. The original COBWEB is not designed to handle uncertain data sets; therefore, *i-CiFHaR* adopts the modified $CU(C_k)$ calculation methodology from the Extended-COBWEB (Xia and Xi, 2007) in order to classify the uncertain mission scenarios according to the pre-computed optimal algorithm hierarchy with the intent of identifying the best matching algorithm cluster.

For example, let F_1 and F_2 represent two attributes and a particular concept cluster, C_1 in the identified tree has three algorithm objects. Let the domain sets of the attributes, F_1 and F_2 be $\{d_{11}, d_{12}\}$ and $\{d_{21}, d_{22}\}$, respectively. Let the objects in the cluster be described in terms of the attribute-value pairs: $[A_1 : \{F_1 = d_{11}, F_2 = d_{21}\}]$, $[A_2 : \{F_1 = d_{11}, F_2 = d_{22}\}]$, and $[A_3 : \{F_1 = d_{11}, F_2 = d_{22}\}]$. Therefore, the concept attribute-value counts are: $[\{F_1 = d_{11} : 3, F_2 = d_{21} : 1, F_2 = d_{22} : 2\}]$. Addressing mission uncertainty, the concept counts include all possible attribute-value pair counts, even when some of the pairs are zero. For example, the concept count is represented as: $[\{F_1 = d_{11} : 3, F_1 = d_{12} : 0, F_2 = d_{21} : 1, F_2 = d_{22} : 2\}]$. A sample mission, described as a vector of *UCAs* is represented as: $[\{F_1 = d_{11} : 0.8, F_1 = d_{12} : 0.2, F_2 = d_{21} : 0.7, F_2 = d_{22} : 0.3\}]$. During the *category utility* computation with the mission added concept C_1, the probability counts take the format: $[\{F_1 = d_{11} : 3.8, F_1 = d_{12} : 0.2, F_2 = d_{21} : 1.7, F_2 = d_{22} : 2.3\}]$. Once the mission scenario is categorized to a particular algorithm cluster, then the identified cluster contains the most appropriate subset of algorithms for the mission. The cluster provides the action choices for *i-CiFHaR*'s influence diagram, which optimizes and ranks only the algorithms within the identified cluster.

IV.1.2.4 Deployment Unit

The responsibility of the *Deployment Unit* is to evaluate the quality of task-coalition pairs. Given a task's robot coalition, this unit validates whether the coalition meets all the task constraints and requirements, and evaluates the coalition quality. When the *Decision Making Module* selects a single coalition formation algorithm to be the only suitable method, then the algorithm is broadcast to the system robots in order to create robot coalitions. Otherwise, when the *Decision Making Module* selects a subset of the most appropriate algorithms to apply, then the *Deployment Unit* evaluates each of the algorithms' applicability by leveraging several metrics (e.g., traveling distance, coalition utility, expected utility scores); and upon enough confidence selects the one that maximizes the likelihood of successfully completing a mission. When none of the selected algorithms satisfy the mission criteria above a certain threshold, the *Deployment Unit* will request the human supervisor to provide additional mission criteria in order to refine the selection of coalition algorithms. Thus, the *Deployment Unit* acts as a wrapper to facilitate decision support to the human user amidst real-world mission conditions.

	Coalition Formation Algorithms						
	A_1	A_2	A_3	A_4	A_5	A_6	A_7
Taxonomy	**Algorithm Category**						
Features	Greedy	Greedy	Auction	Greedy	Greedy	Greedy	Greedy
F_3	Res	Res	Ser	Res	Res	Res	Res
F_5	None	None	None	Org	SNet	SNet	None
F_6	PReq	PReq	PReq	PReq	No	No	PReq
F_7	No	No	Yes	No	No	No	No
F_8	Res	Res	Ser	Res	Res	Res	Res
F_9	No	No	No	No	No	No	Yes
F_{11}	MC	MC	MU	MU	MU	MU	MT
F_{12}	H	H	H	L	L	L	L
F_{13}	IA	IA	IA	TE	IA	TE	IA
F_{14}	No	No	No	No	No	No	No
F_{15}	Yes	No	No	No	No	No	No
F_{16}	Gr	Gr	Auc	Gr	Gr	Gr	Gr
F_{17}	DC	DC	DC	DC	DC	DC	DC
F_{18}	k	k	None	None	k	None	Sngl

	Coalition Formation Algorithms					
	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}
Taxonomy	**Algorithm Category**					
Features	Greedy	Greedy	Auction	Greedy	Approx	Approx
F_3	Res	Res	Res	Ser	Ser	Ser
F_5	None	None	None	None	None	None
F_6	No	PReq	PReq	PReq	No	No
F_7	No	No	No	No	No	No
F_8	Res	Res	Res	Ser	Ser	Ser
F_9	No	No	No	No	No	No
F_{11}	MT	MU	MU	MU	MU	MU
F_{12}	L	H	L	H	H	H
F_{13}	TE	IA	IA	IA	IA	IA
F_{14}	Yes	No	No	No	No	No
F_{15}	No	No	No	No	No	No
F_{16}	Gr	Gr	Auc	Gr	Approx	Approx
F_{17}	DC	DC	DC	DC	C	C
F_{18}	None	k	Sngl	k	None	None

Table IV.3: Taxonomy Features vs Coalition Formation Algorithms (A_1 - A_{13}). See Table IV.1 for feature domains.

	Coalition Formation Algorithms					
	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}
Taxonomy	**Algorithm Category**					
Features	Auction	Greedy	Greedy	Greedy	Auction	Greedy
F_3	Ser	Ser	Ser	Ser	Ser	Res
F_5	None	SNet	None	None	None	None
F_6	PReq	PReq	Yes	Yes	PReq	PReq
F_7	Yes	No	No	No	No	No
F_8	Ser	Ser	Ser	Ser	Ser	Res
F_9	No	No	Yes	Yes	No	No
F_{11}	MU	MT	MU	MT	MU	MU
F_{12}	H	L	H	H	L	H
F_{13}	TE	TE	TE	TE	IA	IA
F_{14}	No	No	Yes	Yes	No	No
F_{15}	No	No	No	No	Yes	Yes
F_{16}	Auc	Gr	Gr	Gr	Auc	Gr
F_{17}	DC	DC	C	C	DC	DC
F_{18}	None	None	None	None	None	None

Feature Domain Key

Approx:Approximation	k:Bounded Size	Res:Resource-Model
Auc:Auction-based	L:Low Communication	Ser:Service-Model
C:Centralized	MC:Minimize Cost	SNet:Social Network
DC:Decentralized	MT:Maximize Tasks Completed	Sngl: Single Sized Coalitions
Gr:Greedy	MU:Maximize Utility	TE:Time-Extended
H:High Communication	Org:Organization Hierarchy	
IA:Instantaneous	PReq:Prerequisite	

Algorithm Key

A_1:(Shehory and Kraus, 1998)	A_2:(Vig and Adams, 2006b)
A_3: (Vig and Adams, 2006a)	A_4:(Abdallah and Lesser, 2004)
A_5: (Tošić and Agha, 2005)	A_6:(Weerdt et al., 2007)
A_7:(Campbell et al., 2008)	A_8:(Sujit et al., 2008)
A_9:(Service and Adams, 2011a)-Resource Model	A_{10}:(Gerkey and Matarić, 2002)
A_{11}:(Service and Adams, 2011a)-Service Model	A_{12}:(Service and Adams, 2011b)-Approximation
A_{13}:(Service and Adams, 2011a)-Dynamic Programming	A_{14}:(Service et al., 2014)-Simultaneous Descending
A_{15}:(Gaston and desJardins, 2005)	A_{16}:(Koes et al., 2005)
A_{17}:(Ramchurn et al., 2010)	A_{18}:(Shiroma and Campos, 2009)
A_{19}:(Zhang et al., 2010)	

Taxonomy Feature Key

F_3: Agent Capability Model	F_5: Agent Structure
F_6: Inter-Task Constraints	F_7: Task Preemption
F_8: Task Requirement Model	F_9: Intra-Task Constraints
F_{11}: Performance Criterion	F_{12}: Communication Overhead
F_{13}: Task Allocation	F_{14}: Spatial Constraints
F_{15}: Overlapping Coalitions	F_{16}: Algorithm Technique
F_{17}: Algorithm Implementation	F_{18}: Coalition Size Constraint

Table IV.4: Taxonomy Features vs Coalition Formation Algorithms (A_{14} - A_{19}). See Table IV.1 for feature domains.

IV.1.3 Library of Algorithms

Nineteen coalition formation algorithms have been selected to be incorporated into *i-CiFHaR*'s library that are associated with all possible feature-value pairs generated from the taxonomy features. This broad collection of algorithms increases the likelihood of *i-CiFHaR* applying to a wide spectrum of missions based on the taxonomy features. The algorithms have been classified into three major categories (greedy, approximation, and auction-based) and are associated with their respective feature-value pairs corresponding to the fourteen prominent features (see Tables IV.3 and IV.4).

IV.2 Experiments and Results

An experiment assessed the efficiency of selecting appropriate coalition formation algorithm(s) based on multiple mission criteria and constraints. The *i-CiFHaR* framework has been implemented on a Linux platform (Ubuntu-12.04, 64-bit) with an Intel Core i5, 2.30GHz processor using C++ and Qt framework (version 4.8) (Nokia, 2012). The purpose of this experiment is to evaluate the *Decision Making Module*'s ability to choose appropriate coalition formation algorithms. The influence diagram implementation leverages the Netica-C API (NORSYS, 2012), a Bayesian network development software tool that uses a junction tree algorithm to evaluate influence diagrams. An open-source Python implementation of COBWEB (McLellan and Harpstead, 2014) was leveraged to generate the hierarchical cluster tree for the coalition formation algorithms. Twenty-four missions were created and the *hypothesis* is that *i-CiFHaR* will select a set of most suitable algorithm(s) to apply for each mission scenario by maximizing the system's expected utility score.

IV.2.1 Experimental Design

The total number of possible mission scenarios is 124,416. Many of these mission scenarios include feature-value pairs that are not realizable for real-world scenarios. The twenty-four mission scenarios delineated in Tables IV.5 and IV.6 represent the subset of realistic situations that were used to evaluate *i-CiFHaR*'s algorithm selection process.

Each mission scenario has a set of feature-value pair assignments for each prominent feature and a probability value. The twenty-four scenarios were simulated in two ways. First, the domain value assigned to each prominent feature was altered, denoted by the Feature-Value pair assignments (FVP) in Tables IV.5 and IV.6. Second, the uncertainty related to each mission was varied by varying the probability value associated with each feature-value pair, denoted by Pr in Tables IV.5 and IV.6. System users can establish domain and mission appropriate probabilities based on domain knowledge, prior mission deployments, intelligence, etc. Feature domain values are mutually exclusive; therefore, the sum of the feature-value probability assignments is 1. Additionally, the mission scenarios were partitioned into clusters (see Table IV.7), such that within each cluster, certain taxonomy feature(s) were instantiated to constant domain values, while the remaining features were altered in order to distinguish between each of the grouped missions.

Let us consider two mission scenarios, MS_1 and MS_3 in order to illustrate the simulation of two different real-world situations. Both missions are based on the *resource model*, where the robots' capabilities and tasks' requirements are described in terms of resources (e.g., camera, sonar, laser). Therefore, the features *Agent Capability* and *Task Requirements* were assigned to *Resource* with a high probability value, 0.8 for both missions. However, the missions differ in many aspects. MS_1's objective is to maximize utility, thus *Performance* is assigned *Maximize Utility (MU)* with a probability of 0.6. Conversely, MS_3 seeks to minimize cost, thus *Performance* is assigned to *Minimize Cost (MC)* with a high probability of 0.9. Additionally, MS_1 does not require overlapping coalitions, thus *Overlapping* is set to *No* with probability 0.8. MS_3 seeks to reduce resource losses; thus, requiring overlapping coalitions, so *Overlapping* is set to *Yes* with a high likelihood of 0.8.

Each mission scenario differs in terms of mission criteria (defined by feature-value assignments and probability values). An exhaustive set of missions cannot be evaluated, thus the twenty-four missions represent a good subset of potential real-world scenarios focused on the prominent taxonomy features and their respective domain sets. The impact of *const* was assessed by varying its value from 25 to 250, in increments of 25; however, this change in value resulted in no variance in the algorithm rankings (Table IV.9). Therefore, the Utility Calculation algorithm (Algorithm 2) and Equation IV.9 *const* value was set to 100 as a designer choice. The variable, μ in Equation IV.13 was set to 90%, based on designer selection.

IV.2.2 Experimental Results

i-CiFHaR selects a subset of the most appropriate algorithms for each mission scenario by optimizing the expected utility score. Table IV.8 presents all the coalition formation algorithms and the missions for which each algorithm was chosen. Table IV.9 ranks the most appropriate algorithms for each mission scenario, where algorithms are ordered by decreasing expected utility scores and the algorithm with the highest score is deemed the most appropriate. A high expected utility score indicates the corresponding algorithm's ability to satisfy the mission's criteria. The subset of the most applicable algorithm(s) is determined by the cutoff threshold, which is a 90% lower bound of the maximum expected utility score. For instance, *i-CiFHaR* selects Service and Adams' heuristic algorithm (A_9) for MS_1 with the maximum expected utility score of 9034.6 as the most suitable algorithm (see Table IV.9). Additionally, the framework selects two additional algorithms with expected utility scores higher than the threshold, $EU^* = 0.9 \times 9034.6 = 8131.2$.

Taxonomy Features	Mission Scenarios					
	MS_1	MS_2	MS_3	MS_4	MS_5	MS_6
F_3	Res,0.8	Res,0.8	Res,0.8	Res,0.8	Ser,0.6	Res,0.8
F_5	None,0.8	None,0.8	None,0.8	None,0.8	None,0.8	SNet,0.6
F_6	PReq,0.7	PReq,0.7	PReq,0.7	PReq,0.7	PReq,0.7	No,0.7
F_7	No,0.8	No,0.8	No,0.8	No,0.8	No,0.8	No,0.8
F_8	Res,0.8	Res,0.8	Res,0.8	Res,0.8	Ser,0.6	Res,0.7
F_9	No,0.8	No,0.8	No,0.8	No,0.8	No,0.8	No,0.8
F_{11}	MU,0.6	MC,0.9	MC,0.9	MU,0.7	MU,0.7	MU,0.7
F_{12}	H,0.8	H,0.8	H,0.8	H,0.8	L,0.7	L,0.9
F_{13}	IA,0.7	IA,0.7	IA,0.7	IA,0.7	IA,0.7	IA,0.6
F_{14}	No,0.8	No,0.8	No,0.8	No,0.8	No,0.8	No,0.8
F_{15}	No,0.8	No,0.8	Yes,0.8	Yes,0.8	Yes,0.9	No,0.8
F_{16}	Gr,0.7	Gr,0.7	Gr,0.7	Gr,0.7	Auc,0.6	Gr,0.7
F_{17}	DC,0.8	DC,0.8	DC,0.8	DC,0.8	DC,0.8	DC,0.8
F_{18}	k,0.6	k,0.6	k,0.6	None,0.7	None,0.7	k,0.5

Taxonomy Features	Mission Scenarios					
	MS_7	MS_8	MS_9	MS_{10}	MS_{11}	MS_{12}
F_3	Res, 0.8	Ser, 0.7	Res, 0.8	Ser, 0.7	Ser, 0.7	Ser, 0.7
F_5	SNet, 0.6	SNet, 0.8	Org, 0.7	None, 0.7	None, 0.7	None, 0.7
F_6	No, 0.7	No, 0.5	PReq, 0.6	PReq, 0.7	PReq, 0.7	No, 0.6
F_7	No, 0.8	No, 0.8	No, 0.8	Yes, 0.9	Yes, 0.9	No, 0.8
F_8	Res, 0.7	Ser, 0.7	Res, 0.7	Ser, 0.8	Ser, 0.8	Ser, 0.8
F_9	No, 0.8	No, 0.8	No, 0.8	No, 0.8	No, 0.8	No, 0.8
F_{11}	MU, 0.7	MT, 0.6	MU, 0.8	MU, 0.8	MU, 0.8	MU, 0.8
F_{12}	L, 0.9	L, 0.9	L, 0.8	H, 0.8	H, 0.8	H, 0.8
F_{13}	TE, 0.7	TE, 0.7	TE, 0.7	IA, 0.6	TE, 0.8	IA, 0.7
F_{14}	No, 0.8	No, 0.8	No, 0.8	No, 0.8	No, 0.8	No, 0.8
F_{15}	No, 0.8	No, 0.8	No, 0.8	No, 0.8	No, 0.8	No, 0.8
F_{16}	Gr, 0.7	Gr, 0.7	Gr, 0.7	Auc, 0.8	Auc, 0.8	Approx, 0.8
F_{17}	DC, 0.8	DC, 0.8	DC, 0.8	DC, 0.8	DC, 0.8	C, 0.6
F_{18}	None, 0.5	None, 0.6	None, 0.5	None, 0.8	None, 0.8	None, 0.8

Table IV.5: Mission Scenarios (MS_1 - MS_{12}) characterized by Feature-Value Pairs. Each mission is defined in the format (Feature-value assignments (FVP), Assignment Probability (Pr)).

Taxonomy Features	**Mission Scenarios** MS_{13}	MS_{14}	MS_{15}	MS_{16}	MS_{17}	MS_{18}
F_3	Ser, 0.9	Res, 0.8	Ser, 0.7	Ser, 0.7	Res, 0.8	Ser, 0.9
F_5	None, 0.7	None, 0.7	None, 0.7	None, 0.7	None, 0.7	None, 0.7
F_6	PReq, 0.6	No, 0.5	Yes, 0.9	Yes, 0.9	PReq, 0.6	Yes, 0.7
F_7	No, 0.8	No, 0.8	No, 0.8	No, 0.8	No, 0.8	Yes, 0.8
F_8	Ser, 0.9	Res, 0.8	Ser, 0.7	Ser, 0.7	Res, 0.8	Ser, 0.9
F_9	No, 0.8	No, 0.5	Yes, 0.8	Yes, 0.8	No, 0.7	Yes, 0.8
F_{11}	MU, 0.8	MT, 0.9	MT, 0.5	MT, 0.8	MU, 0.6	MU, 0.7
F_{12}	H, 0.8	L, 0.9	H, 0.6	H, 0.6	L, 0.8	H, 0.8
F_{13}	IA, 0.7	TE, 0.6	TE, 0.8	TE, 0.8	IA, 0.7	TE, 0.6
F_{14}	No, 0.8	Yes, 0.6	Yes, 0.8	Yes, 0.8	No, 0.8	Yes, 0.8
F_{15}	No, 0.8	No, 0.8	No, 0.8	No, 0.8	No, 0.8	No, 0.8
F_{16}	Approx, 0.8	Gr, 0.6	Gr, 0.8	Gr, 0.8	Auc, 0.6	Gr, 0.6
F_{17}	C, 0.6	DC, 0.7	C, 0.6	C, 0.6	DC, 0.7	DC, 0.5
F_{18}	k, 0.6	None, 0.5	None, 0.6	None, 0.6	Sngl, 0.8	None, 0.7
Taxonomy Features	**Mission Scenarios** MS_{19}	MS_{20}	MS_{21}	MS_{22}	MS_{23}	MS_{24}
F_3	Ser, 0.9	Ser, 0.9	Ser, 0.9	Ser, 0.9	Res, 0.8	Res, 0.8
F_5	None, 0.7	None, 0.7	SNet, 0.6	None, 0.7	None, 0.7	SNet, 0.6
F_6	Yes, 0.6	PReq, 0.7	PReq, 0.7	No, 0.5	No, 0.5	No, 0.7
F_7	Yes, 0.8	Yes, 0.8	Yes, 0.9	No, 0.7	No, 0.7	No, 0.8
F_8	Ser, 0.9	Ser, 0.9	Ser, 0.9	Ser, 0.9	Res, 0.8	Res, 0.8
F_9	Yes, 0.8	Yes, 0.8	No, 0.7	No, 0.7	No, 0.7	No, 0.8
F_{11}	MT, 0.7	MU, 0.7	MT, 0.7	MU, 0.7	MT, 0.5	MT, 0.5
F_{12}	H, 0.8	L, 0.6	L, 0.6	L, 0.7	L, 0.6	L, 0.9
F_{13}	TE, 0.6	TE, 0.5	TE, 0.7	IA, 0.7	TE, 0.6	TE, 0.6
F_{14}	Yes, 0.8	No, 0.7	No, 0.7	No, 0.7	Yes, 0.8	Yes, 0.8
F_{15}	No, 0.8	Yes, 0.9	Yes, 0.8	Yes, 0.9	Yes, 0.8	No, 0.8
F_{16}	Gr, 0.6	Auc, 0.7	Auc, 0.6	Approx, 0.6	Gr, 0.7	Gr, 0.8
F_{17}	DC, 0.5	DC, 0.5	DC, 0.5	DC, 0.5	DC, 0.8	DC, 0.8
F_{18}	None, 0.7	None, 0.5	None, 0.5	None, 0.5	None, 0.5	None, 0.6

Table IV.6: Mission Scenarios ($MS_{13} - MS_{24}$) characterized by Feature-Value Pairs. Each mission is defined in the format (Feature-value assignments (FVP), Assignment Probability (Pr)).

Clustering of Mission Scenarios		
Constant Features	**Impacted Mission Scenarios**	**Features Altered**
F_3, F_5, F_6, F_7, F_8, F_9, F_{13}, F_{14}, F_{15}, F_{17}	MS_1 MS_2 MS_{17}	F_{11}, F_{12}, F_{16}, F_{18}
F_5, F_6, F_7, F_9, F_{13}, F_{14}, F_{15}, F_{17}	MS_3 MS_4 MS_5	F_3, F_8, F_{11}, F_{12}, F_{16}, F_{18}
F_7, F_9, F_{12}, F_{14}, F_{15}, F_{16} F_{17}	MS_6 MS_7 MS_8 MS_9	F_3, F_5, F_6, F_8, F_{11}, F_{13}, F_{18}
F_3, F_5, F_6, F_7, F_8, F_9 , F_{11}, F_{12}, F_{14}, F_{15}, F_{16}, F_{17}, F_{18}	MS_{10} MS_{11}	F_{13}
F_3, F_5, F_7, F_8, F_9 , F_{11}, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, F_{17}	MS_{12} MS_{13}	F_6, F_{18}
F_5, F_7, F_{13}, F_{14}, F_{15}, F_{16}, F_{18}	MS_{14} MS_{15} MS_{16}	F_3, F_6, F_8, F_9 , F_{11}, F_{12}, F_{17}
F_3, F_8, F_{17}, F_{18}	MS_{18} MS_{19} MS_{20} MS_{21} MS_{22}	F_5, F_6, F_7, F_9, F_{11}, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}
F_3, F_6, F_7, F_8, F_9, F_{11}, F_{12}, F_{13}, F_{14}, F_{16}, F_{17}, F_{18}	MS_{23} MS_{24}	F_5, F_{15}

Taxonomy Feature Key

F_3: Agent Capability Model	F_5: Agent Structure
F_6: Inter-Task Constraints	F_7: Task Preemption
F_8: Task Requirement Model	F_9: Intra-Task Constraints
F_{11}: Performance Criterion	F_{12}: Communication Overhead
F_{13}: Task Allocation	F_{14}: Spatial Constraints
F_{15}: Overlapping Coalitions	F_{16}: Algorithm Technique
F_{17}: Algorithm Implementation	F_{18}: Coalition Size Constraint

Table IV.7: Clustering mission scenarios based on features

IV.2.3 Discussion of Results

The experimental results show that *i-CiFHaR* selects appropriate algorithms for each mission scenario. This section discusses the mission scenario results and justifies the selection of the particular algorithms.

Mission scenario 1 (MS_1 in Table IV.5) simulated a mission focused on maximizing total utility that consisted of independent tasks requiring small sized coalitions. All the mission's criteria were satisfied by Service and Adams' heuristic algorithm (A_9) that generates bounded robot coalitions, while maximizing the

	Algorithms	Mission Scenarios
A_1	Shehory and Kraus (1998)	MS_1, MS_2, MS_3, MS_{23}
A_2	Vig and Adams (2006b)	MS_1, MS_2, MS_3, MS_{23}
A_3	Vig and Adams (2006a)	MS_{10}, MS_{11}, MS_{18}, MS_{19}, MS_{20}, MS_{21}
A_4	Abdallah and Lesser (2004)	MS_6, MS_7, MS_9, MS_{24}
A_5	Tošić and Agha (2005)	MS_6, MS_7, MS_{24}
A_6	Weerdt et al. (2007)	MS_6, MS_7, MS_9, MS_{24}
A_7	Campbell et al. (2008)	MS_{14}
A_8	Sujit et al. (2008)	MS_{14}, MS_{23}, MS_{24}
A_9	Service and Adams (2011a)-Resource Model	MS_1, MS_2, MS_3, MS_{17}, MS_{23}
A_{10}	Gerkey and Matarić (2002)	MS_{17}
A_{11}	Service and Adams (2011a)-Service Model	MS_{13}, MS_{18}, MS_{19}, MS_{21}, MS_{22}
A_{12}	Service and Adams (2011b)-Approximation	MS_{12} , MS_{13}, MS_{22}
A_{13}	Service and Adams (2011a)-Dynamic Programming Agent Types	MS_{12} , MS_{13}, MS_{22}
A_{14}	Service et al. (2014)-Simultaneous Descending	MS_{10}, MS_{11}, MS_{18}, MS_{19}, MS_{20}, MS_{21}
A_{15}	Gaston and desJardins (2005)	MS_8, MS_{21}
A_{16}	Koes et al. (2005)	MS_{15}, MS_{16}, MS_{18}, MS_{19}
A_{17}	Ramchurn et al. (2010)	MS_{15}, MS_{16}, MS_{19}
A_{18}	Shiroma and Campos (2009)	MS_5, MS_{20}, MS_{21}, MS_{22}
A_{19}	Zhang et al. (2010)	MS_3, MS_4, MS_5, MS_{23}

Table IV.8: Coalition formation algorithm selections for mission scenarios.

utility. Shehory and Kraus' heuristic algorithm (A_1) and Vig and Adams' algorithm (A_2) were also appropriate (Table IV.9). Both algorithms A_2 and A_9 extend A_1, but A_9 maximizes utility, while A_2 minimizes system cost. The expected utility scores of A_1 and A_2 were similar, because neither can satisfy the mission criterion of maximizing utility.

Mission Scenario 2 (MS_2) was the same as MS_1, except that MS_2 minimized system cost as the performance objective. *i-CiFHaR* selected Shehory and Kraus' heuristic algorithm (A_1) as the best algorithm and Vig and Adams' algorithm (A_2) as the second most suitable algorithm. Both algorithms satisfied all the mission criteria and are associated with the the same feature-value pairs, except overlapping coalitions. A_2 extends A_1 for real-robot domains and both seek to minimize cost. Service and Adams' algorithm (A_9) ranked

Mission Scenarios	**Rankings of Algorithms**				
	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
MS_1	A_9 (9034.6)	A_1 (8374.5)	A_2 (8328.7)	-	-
MS_2	A_1 (8716.7)	A_2 (8669.2)	A_9 (8585.1)	-	-
MS_3	A_1 (8035.1)	A_{19} (7882.9)	A_2 (7784.6)	A_9 (7699.6)	-
MS_4	A_{19} (9047.2)	-	-	-	-
MS_5	A_{18} (7458.3)	A_{19} (7001.4)	-	-	-
MS_6	A_5 (7414.6)	A_6 (7321.3)	A_4 (6983.2)	-	-
MS_7	A_6 (7686.7)	A_4 (7339.3)	A_5 (7046.6)	-	-
MS_8	A_{15} (7047.3)	-	-	-	-
MS_9	A_4 (7906.7)	A_6 (7195.9)	-	-	-
MS_{10}	A_3 (8213.5)	A_{14} (7704.5)	-	-	-
MS_{11}	A_{14} (8175.3)	A_3 (7808.4)	-	-	-
MS_{12}	A_{12} (7658.6)	A_{13} (7658.6)	-	-	-
MS_{13}	A_{11} (7737.2)	A_{12} (7142.9)	A_{13} (7142.9)	-	-
MS_{14}	A_8 (6075.0)	A_7 (5644.9)	-	-	-
MS_{15}	A_{16} (5531.8)	A_{17} (5183.1)	-	-	-
MS_{16}	A_{17} (5441.5)	A_{16} (5268.1)	-	-	-
MS_{17}	A_{10} (7476.8)	A_9 (7190.2)	-	-	-
MS_{18}	A_{16} (5336.9)	A_{14} (5031.2)	A_3 (4960.9)	A_{11} (4890.3)	-
MS_{19}	A_{17} (4679.5)	A_{16} (4621.8)	A_{14} (4620.9)	A_3 (4554.4)	A_{11} (4480.6)
MS_{20}	A_{18} (5287.2)	A_3 (4960.9)	A_{14} (4791.4)	-	-
MS_{21}	A_{15} (4491.5)	A_{14} (4439.5)	A_{18} (4390.1)	A_3 (4237.3)	A_{11} (4143.8)
MS_{22}	A_{18} (5908.2)	A_{12} (5337.5)	A_{13} (5337.5)	A_{11} (5331.2)	-
MS_{23}	A_8 (5444.8)	A_{19} (5370.2)	A_1 (5241.5)	A_2 (5023.7)	A_9 (5002.2)
MS_{24}	A_6 (6734.9)	A_8 (6650.5)	A_4 (6367.6)	A_5 (6083.8)	-

Table IV.9: Ranking of coalition formation algorithms by decreasing expected utility scores for each mission scenario. Each mission describes the most appropriate subset of algorithms in the format, *Algorithm (Expected Utility Score)*.

third, because it satisfied all mission requirements, but the cost minimization requirement.

Mission Scenario 3 (MS_3) required overlapping and bounded coalitions, while minimizing the overall system cost. Four algorithms were selected with Shehory and Kraus' algorithm (A_1) ranked as the most appropriate. A_1 is the only algorithm that allows overlapping and bounded coalitions, while also minimizing system cost. Despite generating overlapping coalitions, Zhang et al.'s algorithm (A_{19}) was selected as the second choice, because it maximizes total utility. Vig and Adams' algorithm (A_2) caters to almost all the mission criteria, but does not permit overlapping coalitions, despite being an extension of A_1. Service and Adams' algorithm (A_9) ranked fourth, because it does not permit overlapping coalitions and maximizes the total utility.

Mission Scenario 4 (MS_4) required overlapping coalitions and the maximization of the system utility, but did not restrict the coalition sizes. *i-CiFHaR* selected Zhang et al.'s algorithm (A_{19}) as the only best fit algorithm for the mission. The algorithm leverages particle swarm based optimization technique in order to generate overlapping coalitions, while maximizing system utility without incorporating the bounded coalition size heuristic.

The communication bandwidth availability was high for both MS_3 and MS_4; however, Mission Scenario 5 (MS_5) required a low communication footprint due to constrained bandwidth. Moreover, MS_5 required overlapping coalitions and services (e.g., box-pushing, foraging, sentry-duty) were used to represent the agents' capabilities and the tasks' requirements. Shiroma and Campos' CoMuTaR (A_{18}) is a service-based coalition formation algorithm that permits overlapping coalitions and requires low communication bandwidth, as a result it was ranked as the most appropriate algorithm. Zhang et al.'s algorithm (A_{19}) was selected as the second most appropriate alternative, but it does not satisfy two mission criteria, namely the low communication and the service model-based requirements.

Consider the first response example from Section 1, where coalitions of robots assess the situation. Such real-world environments often require low communications. Mission Scenario 6 (MS_6) depicts such a situation, where the communication bandwidth is restricted due to environmental constraints and small-sized coalitions, bounded within a maximum limit of k are preferred. Moreover, the critical situation demanded instantaneous task allocations with robots forming a social network topology based on the limited inter-robot communication, resulting in a sparse network. Given the mission criteria, *i-CiFHaR* ranked Tošić and Agha's algorithm (A_5) as the most appropriate, because it leverages a social network and requires low communication bandwidth, when the underlying topology graph is sparse. Weerdt et al.'s algorithm (A_6) was ranked second, because it too leverages a team of social networked robots to compute task coalitions under constrained communication requirements. Abdallah and Lesser's heuristic algorithm (A_4) satisfies almost all the mission criteria and it was ranked third, because it considers robots to be part of an organization hierarchy, rather than a social network.

Mission Scenario 7 (MS_7) was very similar to MS_6, except that MS_7 required time-extended allocations that stem from the need for task scheduling. Additionally, there was no restrictions on the coalition sizes. Given the new conditions, *i-CiFHaR* ranked Weerdt et al.'s algorithm (A_6) as the most appropriate algorithm, because it provides time-extended task allocations with a low communication overhead and leverages a social network. Abdallah and Lesser's algorithm (A_4) ranked second, because it permits time-extended allocations with limited communication bandwidth requirements and unbounded coalitions. Tošić and Agha's algorithm (A_5) was ranked third, because it did not meet the time-extended mission requirement, while it computes only bounded coalitions.

Mission Scenario 8 (MS_8) differed from MS_6 and MS_7, but the robots' communication structure remained the same (Social network). This mission was different in that it required a service-based model and the objective was to maximize the number of tasks to be completed within a stipulated time frame, unlike the utility maximization objective of MS_6 and MS_7. Time-extended allocation was also a criterion. Gaston and desJardins' coalition formation algorithm (A_{15}) was selected as the most appropriate algorithm that satisfied all the mission requirements.

Mission Scenario 9 (MS_9) was similar to MS_6, but simulated a scenario where robots are connected in an organizational hierarchy. Low communication was a mission constraint and there was no bound on the coalition size. Time-extended allocations were necessary. *i-CiFHaR* selected Abdallah and Lesser's algorithm (A_4) as the most appropriate algorithm, because this is the only algorithm that utilizes an organizational hierarchy to compute time-extended coalitions with no size restrictions. Weerdt et al.'s algorithm (A_6) ranked second.

Mission scenario 10 (MS_{10}) included a number of high priority tasks requiring frequent task preemption. MS_{10} required a *service model* and an auction-based coalition formation algorithm. Vig and Adams' RACHNA (A_3) ranked as the most appropriate algorithm, because it satisfied all mission requirements, including task preemption. Service et al.' simultaneous descending auction-based algorithm (A_{14}), an extension of RACHNA, was ranked second.

Mission Scenario 11 (MS_{11}) required task preemption and a service-model similar to MS_{10}; however, MS_{11} required time-extended allocations. *i-CiFHaR* selected the simultaneous descending auction algorithm (A_{14}) as the most appropriate algorithm, since it generates both instantaneous and time-extended coalitions, while permitting online task preemption. Vig and Adams' RACHNA (A_3) ranked second, because it allows only instantaneous allocations.

Mission Scenario 12 (MS_{12}) simulated critical tasks requiring high utility coalitions with guaranteed solution quality, thereby requiring approximation algorithms. Agent capabilities and task resource requirements were expressed in terms of services (e.g., patrolling, sentry-duty, foraging), while robot coalition sizes were not constrained. *i-CiFHaR* selected the two approximation algorithms, A_{12} and A_{13} with equal expected utility scores. The two algorithms are equally applicable to MS_{12}, since they are associated with the same feature-value pairs.

Mission Scenario 13 (MS_{13}) differed from MS_{12} in that coalition sizes were restricted by an upper bound, k and solution quality was not critical; therefore, there was an equal likelihood that either a greedy or an approximation algorithm is applicable. The remaining mission requirements were identical to MS_{12}. Service and Adams' service-model based heuristic algorithm (A_{11}) was selected the most appropriate algorithm, since it met all mission requirements. *i-CiFHaR* selected the two approximation algorithms, A_{12} and A_{13} as the remaining alternatives.

Real robots must travel to the assigned task's location; therefore, satisfying spatial constraints in real-world conditions is essential. Mission Scenario 14 (MS_{14}) required that robot coalitions met spatial constraints and had a low communication footprint. Time-extended task allocations were necessary and the performance objective was to maximize the number of tasks completed in a given time. *i-CiFHaR* selected Sujit et al.'s algorithm (A_8) as the most suitable fit, because this algorithm satisfies spatial constraints using Dubin's curves to estimate the travel time to task locations. Moreover, A_8 uses low inter-robot communication bandwidth ($O(n \times m)$) with n robots and m tasks, and maximizes the number of tasks completed. Campbell et al.'s heuristic algorithm (A_7) ranked second, since it creates single robot coalitions with no inter-agent communication at all, and models the coalition formation problem as a multi-processor scheduling problem to maximize the number of completed tasks.

Mission Scenario 15 (MS_{15}) simulated tasks with hard task completion deadlines. Additionally, MS_{15} involved tasks with dependencies, invoking the need to satisfy inter-task precedence constraints. Consider two mission tasks: the first is to triage a victim and the second is to clear debris covering an injured victim. The second task must be performed first. The performance objective of MS_{15} was to maximize system utility, while time-extended allocations were preferred. *i-CiFHaR* selected Koes et al.'s algorithm (A_{16}) as the best, because it met all mission requirements. Ramchurn et al.'s algorithm (A_{17}) was second, because it maximizes the number of completed tasks.

Mission Scenario 16 (MS_{16}) differed from MS_{15} in that its performance objective was to maximize the number of completed tasks, while the remaining mission requirements remained the same. *i-CiFHaR* selected Ramchurn et al.'s algorithm (A_{17}) as the most appropriate algorithm, as it satisfied all mission criteria. This time Koes et al.'s algorithm (A_{16}) was selected as the second most suitable alternative.

Mission scenario 17 (MS_{17}) depicted a situation where single robot coalitions were necessary under low communication requirements, and auction-based algorithms were preferred. Gerkey and Matarić's MUR-

DOCH (A_{10}) ranked as the best fit, followed by Service and Adams' resource-model based algorithm (A_9). The latter was ranked lower, because it requires high inter-agent communication messaging for computing coalitions.

Mission situation MS_{18} was a conglomeration of several critical requirements, including task preemption and addressing of inter- and intra-task constraints along with spatial constraints. Additionally, the mission required scheduling, thus time-extended allocation was crucial for the team of service-based robots. *i-CiFHaR* considered all the critical criteria and selected Koes et al.'s centralized coalition formation algorithm (A_{16}), because this algorithm addresses both inter- and intra-task constraints, alongside spatial constraints. Moreover, Koes et al.'s algorithm permits time-extended allocations, but fails to allow task preemption. Service et al.'s simultaneous descending auction approach (A_{14}) was ranked second, on the grounds that this algorithm allows task preemption and time-extended allocations; however, fails to address the task and spatial constraints. RACHNA (A_3) being a task preemptive approach was ranked third, followed by Service and Adams' service-model algorithm (A_{11}) as the fourth alternative.

The mission requirements of the next scenario (MS_{19}) were almost identical to that of MS_{18}, except that the former sought to maximize the number of completed tasks that were spatially distributed. *i-CiFHaR* re-evaluated the requirements and recommended Ramchurn et al.'s coalition formation algorithm (A_{17}), because it satisfies most of the mission criteria, except task preemption. Koes et al.'s algorithm (A_{16}) was ranked second, because it aims to maximize the utility and does not permit preemption; however, it does satisfy the remaining criteria. Service et al.'s approach (A_{14}) was ranked higher than RACHNA (A_3), because the former allows time-extended allocations along with task preemption, whereas the latter permits instantaneous allocations and task preemption. Service and Adams' service-model algorithm (A_{11}) was the fifth alternative.

Mission Scenario 20 (MS_{20}) demanded two additional criteria. Aiming to reduce resource usage, the mission preferred overlapping coalitions. Low communication bandwidth was required alongside the aforementioned criteria of MS_{18} and MS_{19}. CoMutaR (A_{18}) was ranked first, owing to the fact that this service-model based algorithm computes overlapping coalitions with low communication bandwidth requirements. RACHNA (A_3) and Service et al.'s algorithm (A_{14}) were chosen as the next suitable alternatives, because these algorithms allow task preemption and seek to maximize the utility.

The robots in mission scenario MS_{21} were connected in a communication social network and task preemption and time-extended allocations were necessary. The objective was to maximize the number of completed tasks and reduce the communication footprint. *i-CiFHaR* selected Gaston and desJardins' service-model algorithm (A_{15}) that employs a social network as a heuristic to perform the time-extended task allocations with reduced communication overhead. The simultaneous descending auction algorithm (A_{14}) allowing task preemption and time-extended allocations ranked second. CoMutaR (A_{18}) ranked third and falls short of maximizing the completed tasks and task preemption objective criteria. RACHNA (A_3) ranked fourth, given the fact that despite permitting task preemption, A_3 maximizes system utility and have instantaneous allocations. Service and Adams' service-model algorithm (A_{11}) was the fifth alternative.

Mission scenario MS_{22} aimed for approximate overlapping coalitions along with low communication overhead and the objective to maximize team utility. *i-CiFHaR* selected CoMutaR (A_{18}), because this algorithm allows overlapping coalitions with constraints on communication bandwidth and maximizes total team utility. However, the approach is an auction-based technique and fails to provide guarantees on the solution quality. *i-CiFHaR* recommended Service and Adams' two approximation algorithms, A_{12} and A_{13} as the next two highest ranked algorithms, because they provide solutions within a fixed bound from the optimal solutions. However, the approximation algorithms fail to allow overlapping coalitions with a low communication footprint. Service and Adams' service-model algorithm (A_{11}) was the fourth alternative.

Mission scenario MS_{23} was designed for robots that follow the *Resource-Model*. The mission requirements involved overlapping coalitions with time-extended allocations for spatially distributed tasks. *i-CiFHaR* selected Sujit et al.'s coalition formation algorithm (A_8) that uses Dubin's curves to address spatially distributed tasks and computes time-extended allocations with low communication messages. However, this algorithm does not permit overlapping coalitions. Zhang et al.'s particle-swarm based approach (A_{19}) and Shehory and Kraus' algorithm (A_1) were selected as the second and third choices respectively, since both algorithms allow overlapping coalitions, but fail to provide time-extended allocations. Vig and Adams' algorithm (A_2), which is an extension of A_1 was ranked fourth, while Service and Adams' algorithm (A_9) was the last alternative.

The last mission, MS_{24} was similar to MS_{23}, except that the robots used a communication topology as a social network. *i-CiFHaR* selected Weerdt et al.'s algorithm (A_6) as the most appropriate algorithm, based on the fact that this algorithm offers time-extended allocations with low communication footprints and leverages a social network to compute coalitions. Sujit et al.'s approach (A_8) was ranked second, because it seeks to maximize the number of completed tasks and provides time-extended allocations with low communication overhead, few of the mission criteria. A_4 and A_5 were the remaining alternatives.

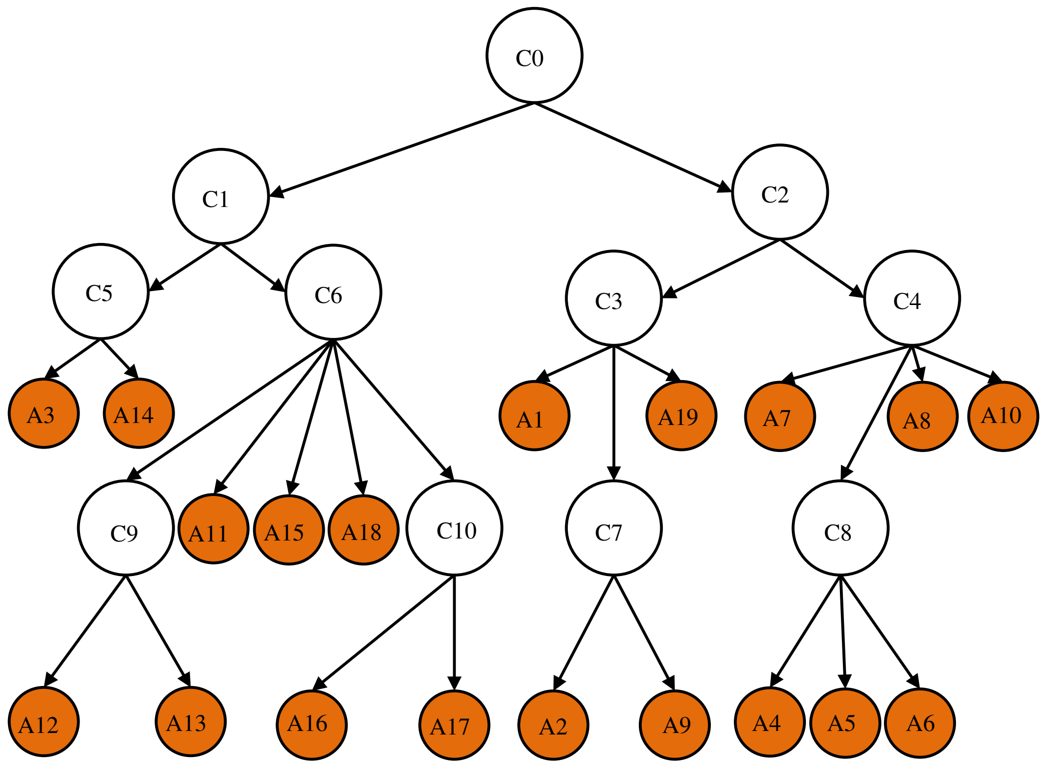
The results show that for all the simulated mission scenarios, *i-CiFHaR* successfully selected the most appropriate algorithms to apply based on multiple mission criteria. When a single best fit algorithm is not found, *i-CiFHaR* determines a subset of algorithms that are most suitable. The highlighted missions represent a subset of all possible scenarios; however, the number of all possible missions is exorbitantly large. The selected mission scenarios exploited all taxonomy features and provided a good subset of the possible missions.

IV.2.4 Experimental Results for *i-CiFHaR* with Clustering

The hierarchical classification tree with the maximum PU score, as identified by COBWEB, is shown in Table IV.10. The hierarchy is represented by the levels with the root concept, $C0$ containing all the coalition formation algorithms at Level-0. As one moves down the hierarchy, children $C1$ and $C2$ at Level-1 partition the entire library into $Service - model$ and $Resource - model$ based algorithms. More concept clusters are realized that group similar algorithms based on their attribute-value pairs lower in the hierarchy levels. The colored leaf nodes represent the nineteen singleton concepts, each one corresponding to a single coalition formation algorithm.

i-CiFHaR acts as a decision support system; therefore, it selects either a single coalition formation algorithm, or a subset of algorithm(s) that satisfy all or most of a given mission's criteria. *i-CiFHaR* analyzes the most suitable cluster, as identified by COBWEB and optimizes the algorithm rankings by maximizing the expected utility score. The coalition formation algorithm rankings for each of the twenty four missions are provided in Table IV.11. The table also provides the pertinent cluster and the cluster size by mission.

Figure IV.5 presents *i-CiFHaR*'s computational time with and without the COBWEB clustering. It is noted that *i-CiFHaR*'s computational time with clustering is lower than *i-CiFHaR* without clustering across all the mission scenarios. The mean time of the latter is 16.1 seconds, with a standard deviation is 0.2 seconds. Conversely, *i-CiFHaR* with clustering computed solutions in a mean time of 5.2 seconds, with a standard deviation of 2.7 seconds, a 67% improvement. The computational time standard deviation differences stems from the fact that *i-CiFHaR* without clustering always considered all nineteen algorithms, irrespective of the mission scenario. The conceptual clustering based *i-CiFHaR* selects a single best cluster of algorithms to apply to a given mission, but the cluster sizes differ, as shown in Table IV.11. The computational time of certain missions (e.g., MS_{19}, MS_{20}) is much higher than that of other missions (e.g., MS_1, MS_3, MS_{11}), because the cluster sizes in the former scenarios are much larger than those of the latter.



Algorithm Key

A_1:(Shehory and Kraus, 1998)	A_2:(Vig and Adams, 2006b)
A_3: (Vig and Adams, 2006a)	A_4:(Abdallah and Lesser, 2004)
A_5: (Tošić and Agha, 2005)	A_6:(Weerdt et al., 2007)
A_7:(Campbell et al., 2008)	A_8:(Sujit et al., 2008)
A_9:(Service and Adams, 2011a)-Resource Model	A_{10}:(Gerkey and Matarić, 2002)
A_{11}:(Service and Adams, 2011a)-Service Model	A_{12}:(Service and Adams, 2011b)-Approximation
A_{13}:(Service and Adams, 2011a)-Dynamic Programming	A_{14}:(Service et al., 2014)-Simultaneous Descending
A_{15}:(Gaston and desJardins, 2005)	A_{16}:(Koes et al., 2005)
A_{17}:(Ramchurn et al., 2010)	A_{18}:(Shiroma and Campos, 2009)
A_{19}:(Zhang et al., 2010)	

Table IV.10: The hierarchical cluster tree generated by conceptual clustering. The white circles constitute the cluster concepts, while the colored circles or leaf nodes represent the singleton concepts containing a single algorithm.

i-CiFHaR with clustering produced identical algorithm rankings as *i-CiFHaR* without clustering (see Tables IV.9 and IV.11) for twenty-three mission scenarios. *i-CiFHaR* without clustering selected two algorithms, A_{18} and A_{19} for MS_5; however, *i-CiFHaR* with clustering chose only *Service-model* based algorithms, A_{18} from cluster $C6$ and excluded A_{19}, because A_{19} leverages a *Resource-model* and belongs to cluster $C3$.

i-CiFHaR's computational time, by cluster size is provided in Table IV.12. *i-CiFHaR*'s computational time with COBWEB increases linearly with the cluster size. However, as *i-CiFHaR* scales to include more algorithms in the

Cluster # (Size)	**Mission Scenarios**	**Rankings of Algorithms**				
		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
$C3(4)$	MS_1	A_9	A_1	A_2	-	-
$C3(4)$	MS_2	A_1	A_2	A_9	-	-
$C3(4)$	MS_3	A_1	A_{19}	A_2	A_9	-
$C3(4)$	MS_4	A_{19}	-	-	-	-
$C6(7)$	MS_5	A_{18}	-	-	-	
$C8(3)$	MS_6	A_5	A_6	A_4	-	-
$C8(3)$	MS_7	A_6	A_4	A_5	-	-
$C6(7)$	MS_8	A_{15}	-	-	-	-
$C8(3)$	MS_9	A_4	A_6	-	-	-
$C5(2)$	MS_{10}	A_3	A_{14}	-	-	-
$C5(2)$	MS_{11}	A_{14}	A_3	-	-	-
$C9(2)$	MS_{12}	A_{12}	A_{13}	-	-	-
$C6(7)$	MS_{13}	A_{11}	A_{12}	A_{13}	-	-
$C4(6)$	MS_{14}	A_8	A_7	-	-	-
$C10(2)$	MS_{15}	A_{16}	A_{17}	-	-	-
$C10(2)$	MS_{16}	A_{17}	A_{16}	-	-	-
$C2(10)$	MS_{17}	A_{10}	A_9	-	-	-
$C1(9)$	MS_{18}	A_{16}	A_{14}	A_3	A_{11}	-
$C1(9)$	MS_{19}	A_{17}	A_{16}	A_{14}	A_3	A_{11}
$C1(9)$	MS_{20}	A_{18}	A_3	A_{14}	-	-
$C1(9)$	MS_{21}	A_{15}	A_{14}	A_{18}	A_3	A_{11}
$C6(7)$	MS_{22}	A_{18}	A_{12}	A_{13}	A_{11}	-
$C2(10)$	MS_{23}	A_8	A_{19}	A_1	A_2	A_9
$C4(6)$	MS_{24}	A_6	A_8	A_4	A_5	-

Table IV.11: Algorithm rankings of each mission scenario by decreasing expected utility scores. The cluster sizes for each mission are also provided.

library, COBWEB can generate a different classification hierarchy tree comprised of clusters with different sizes. An increased cluster size will result in increased computational time. The worst case leverages the root cluster containing all algorithms in the library, as is utilized by *i-CiFHaR* without clustering; thereby, considering $O(n)$ algorithms for decision making. However, with the hierarchical clustering approach, *i-CiFHaR* potentially will leverage $O(\log_b n)$ algorithms, where n is the number of algorithms and b is the average branching factor of the hierarchical tree.

Cluster Size	2	3	4	6	7	9	10	19
Mean Computation Time (sec)	2	2.5	3.5	5.5	6.8	8.7	9.8	16.1

Table IV.12: Cluster Size vs Average Computation Time (sec).

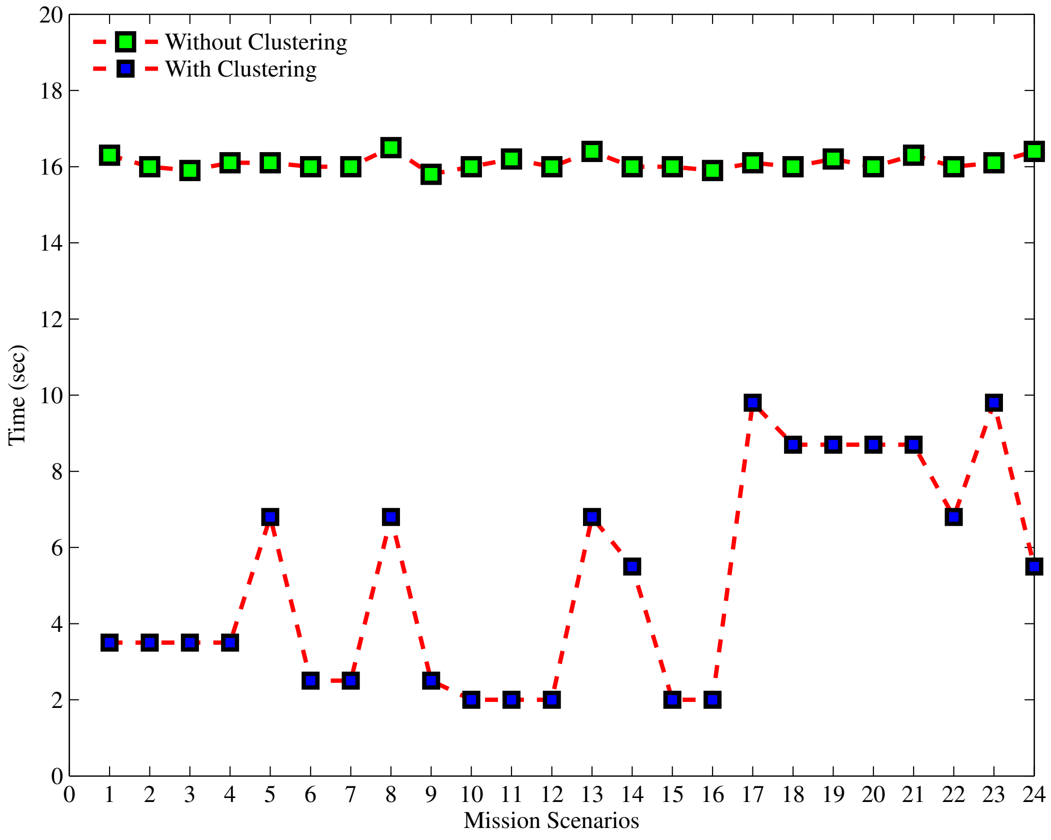
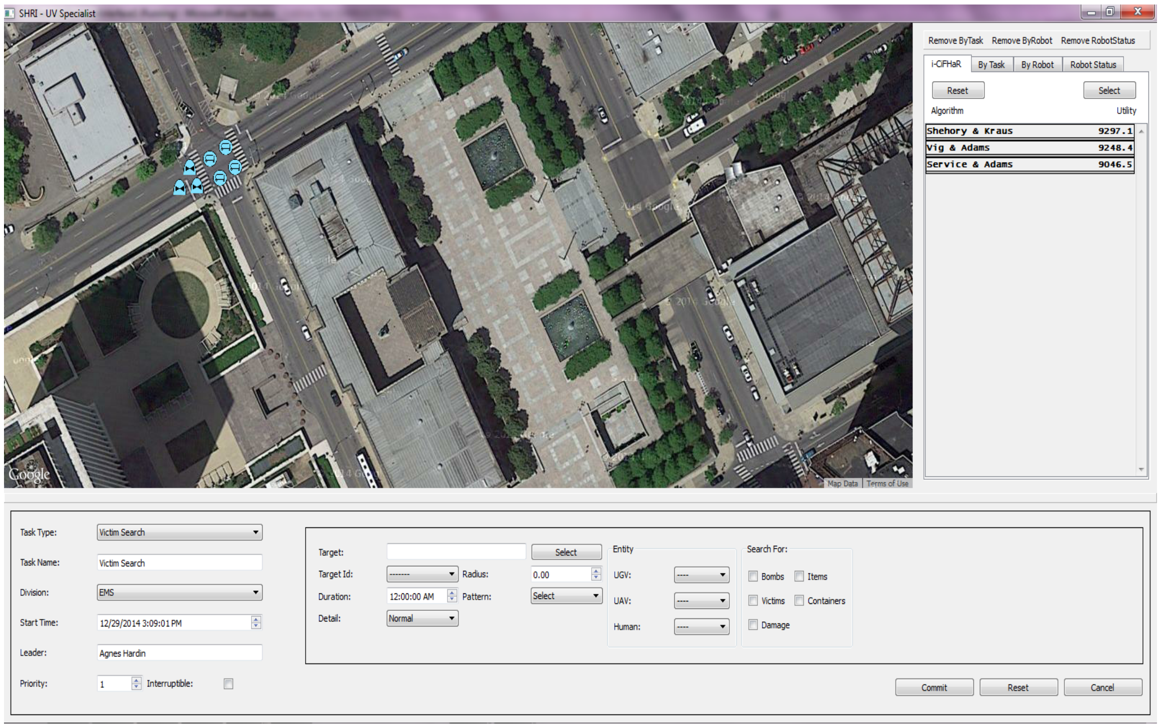


Figure IV.5: *i-CiFHaR*'s computational time in seconds with and without conceptual clustering.

IV.2.5 Human Interface Integration Results

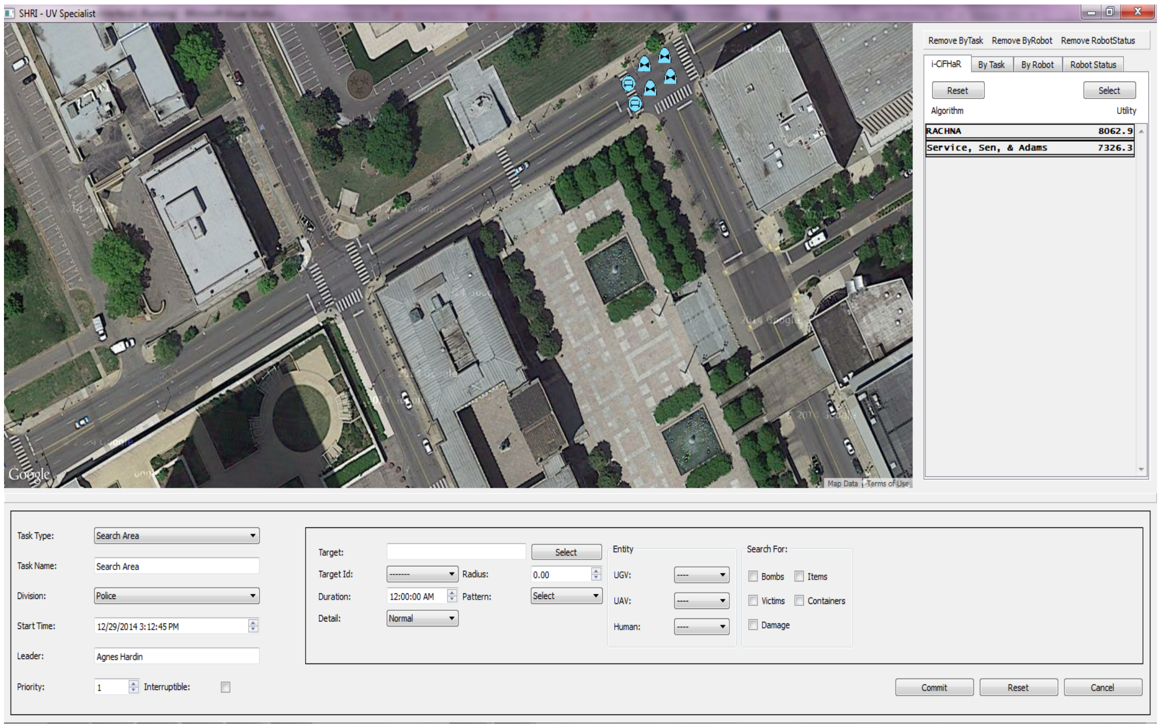
The *i-CiFHaR*'s middle level logic tier leverages an influence diagram for performing optimized selection of the most appropriate coalition formation algorithms to apply to given mission scenarios with multiple mission criteria. Aiming to develop *i-CiFHaR* into a complete decision support system for mission supervisors, the middle level logic tier, along with the library of algorithms are integrated into the existing Human Machine Teaming Laboratory's SHRI (System of Human Robot Interface) module. SHRI uses Google Maps to provide the user (mission specialist) with the ability to specify missions. The user inputs a number of mission criteria, such as task type, task name, task duration, priority, required types of unmanned vehicles, etc. All these criteria are parsed into an XML file and sent to *i-CiFHaR*, which reads the file to extract relevant information for the decision making. *i-CiFHaR* performs the probabilistic reasoning and derives the most suitable subset of coalition formation algorithms based on its library. This list of selected algorithms, along with their ranks and expected utility scores are parsed into an XML file and sent to the SHRI, which in turn provides the resulting coalition rankings to allow the user to choose the most appropriate coalition for the mission. Figures IV.6 and IV.7 display two sets of most suitable algorithms for two separate missions, *Victim Search* and *Search Area*, respectively. The two tasks required the same set of criteria similar to that of Mission Scenario 2 and Mission Scenario 10 from the *i-CiFHaR* experiments (see Table IV.5). The algorithm rankings are displayed in the right panel of the interface. It is noted that the integration of the interface is a proof of concept for providing decision support to the human operator.



Algorithm	Utility
Shehory & Kraus	9297.1
Vig & Adams	9248.4
Service & Adams	9046.5

Task Type: Victim Search
Task Name: Victim Search
Division: EMS
Start Time: 12/29/2014 3:09:01 PM
Leader: Agnes Hardin
Priority: 1 Interruptible:
Target:
Target Id: -------
Duration: 12:00:00 AM
Detail: Normal
Radius: 0.00
Pattern: Select
UGV: ----
UAV: ----
Human: ----

Figure IV.6: The SHRI GUI for a *victim search* task with algorithm ranking in the right panel.



Algorithm	Utility
RACHNA	8062.9
Service, Sen, & Adams	7326.3

Task Type: Search Area
Task Name: Search Area
Division: Police
Start Time: 12/29/2014 3:12:45 PM
Leader: Agnes Hardin
Priority: 1 Interruptible:
Target:
Target Id: -------
Duration: 12:00:00 AM
Detail: Normal
Radius: 0.00
Pattern: Select
UGV: ----
UAV: ----
Human: ----

Figure IV.7: The SHRI GUI for a *search area* task with algorithm ranking in the right panel.

IV.3 Chapter Summary

An intelligent framework is presented that reasons online over a library of coalition formation algorithms to select the most appropriate coalition formation algorithm(s) to apply to a given mission scenario, and result in a coalition to execute the mission. *i-CiFHaR* reasons using a suite of algorithms, rather than a single heuristic based algorithm and leverages an influence diagram to make decisions online under multiple, uncertain mission criteria. A link analysis based algorithm calculates the utility values of the feature-value pairs. The framework uses a number of features to select the most suitable coalition formation algorithm(s). The curse of dimensionality is addressed by extracting prominent features that discriminate the coalition algorithms using principal component analysis. These prominent features are utilized to dynamically create the influence diagram at run-time. The experimental results show that *i-CiFHaR* selects the appropriate algorithm(s), given multiple mission criteria. The framework additionally incorporates conceptual clustering in order to cluster similar algorithms in its library. Based on the clusters, *i-CiFHaR* selects the most pertinent partition for a given mission, and ranks the algorithm in this identified cluster. The computational time of *i-CiFHaR* is improved by 67% through the use of the clustering mechanism. When a single best fit algorithm is unavailable, *i-CiFHaR* selects a subset of suitable algorithms that are applicable to form coalitions. *i-CiFHaR* is applicable to missions with frequent contingency occurrences that introduce changing mission requirements (e.g., overlapping coalitions resulting from robot failures, task preemption). The likelihood of handling diverse situations increases with the inclusion of a broad set of algorithms in the system. *i-CiFHaR* provides a more robust approach to allocate task coalitions for dynamic, real-world scenarios. An existing human-computer interface is integrated into *i-CiFHaR* in order to provide decision support to human operators.

CHAPTER V

Conclusions, Contributions, and Future Work

This chapter summarizes the dissertation and outlines the major contributions that stem from this research. A number of potential directions for future work are also provided.

V.1 Dissertation Summary

This dissertation presented and validated *i-CiFHaR*, a novel coalition formation framework that leverages influence diagram to perform probabilistic reasoning regarding the most appropriate coalition algorithm to apply to a given mission. *i-CiFHaR* is a first of its kind system that incorporates a library of coalition formation algorithms instead of the contemporary use of a single algorithm. The framework incorporates conceptual clustering in order to partition the algorithms in its library; thereby, selecting only the most suitable cluster of algorithms for application analysis. The ranking of the algorithms in this identified cluster is accomplished by optimizing the expected utility score.

Twenty four mission scenarios were used to evaluate the effectiveness of *i-CiFHaR*'s decision making process both with and without the integration of the clustering mechanism. *i-CiFHaR* selected the best fit algorithms for each of the missions and the clustering approach further corroborated the results, since the algorithms selected by *i-CiFHaR* without clustering matched those that were obtained through the use of clustering. The integration of the conceptual clustering technique improved *i-CiFHaR*'s computation time by 67%.

Conventional coalition formation algorithms are classified into greedy, market-based, and approximation categories; however, each of the algorithm classes has its own pros and cons during its application to multi-robot systems. Many of the existing heuristic coalition formation approaches leverage the heuristic of constraining the coalition sizes upto a maximum limit of $k << n$ (n = number of agents/robots) in an attempt to compute good coalitions in real-time. This detrimental heuristic considerably affects the coalition quality, because a small k value may result in poor coalitions due to insufficient exploration of the search space. This dissertation attempts to bridge the gap by leveraging an ACO-based coalition formation algorithm that can compute solutions in real-time for very large teams of robots, without the use of any detrimental heuristics. However, existing ACO algorithms, popular for solving combinatorial optimization problems suffer from search stagnation, which limits the extensiveness of the search process. Therefore, this dissertation addresses this basic shortcoming of ACO algorithms by contributing two novel pheromone depositing policies by integrating simulated annealing technique; thereby, leading to the development of two centralized generic and improved ACO search algorithms (*sA-ANT* and *sA-ANT**) that are applicable to any *NP*-complete problems.

The centralized *sA-ANT* and *sA-ANT** algorithms were applied to three optimization problems: Traveling Salesman Problem, multi-agent coaltion formation problem, and the maximal clique generation problem. The experimental results illustrate the effectiveness of the presented hybrid algorithms in addressing the search stagnation drawback by exhibiting a significant improvement in their search exploration capabilities. This enhanced searching resulted in *sA-ANT** and *sA-ANT*s' significantly improved solution qualities for the TSP and coalition formation problem when compared to existing state-of-the-art ACO-based approaches. *sA-ANT** performed better than *Ant-Clique* algorithm for the Maximal Clique problem, although the improvement was not statistically significant.

A true multi-robot coalition formation algorithm is required to be decentralized for robustness to dynamic real-world situations and fault-tolerance. Therefore, distributed *sA-ANT** (*d-sA-ANT**) is presented, which permits a team of robots to concurrently compute coalitions in real-time in a distributed fashion. *d-sA-ANT** incorporates information sharing across the robots by allowing each robot to publish world information using its immediate neighbors; thereby, resulting in an information propagation through multicast message routing. Each robot employs its own set of ants, as multiple agents to perform the search process. Experimental results demonstrate that *d-sA-ANT** computes coalitions of almost identical qualities as those generated by the centralized *sA-ANT** algorithm; however, the computation time of the former is significantly higher than that of the centralized variant, because of the information propagation overhead.

V.2 Contributions

This dissertation contributes to the multi-agent coalition formation and swarm intelligence communities. The presented work improves upon the state-of-the-art in both fields as stated subsequently.

1. The primary contribution to the multi-agent systems community is the development of an intelligent coalition formation framework, *i-CiFHaR*, which performs probabilistic reasoning to compute the most appropriate coalition formation algorithm(s) to create robust coalitions for real-time missions. The framework is the first of its kind that leverages a library of diverse algorithms, rather than a single algorithm. *i-CiFHaR* leverages an influence diagram/decision network to optimize the algorithm selection process by maximizing the expected utility score. The inclusion of a broad set of diverse coalition formation algorithms renders *i-CiFHaR* flexible and applicable to a wide-range of real-world missions and increases the likelihood of mission success. *i-CiFHaR* can provide efficient decision support to human mission supervisors amidst immense stresses associated with field deployments.

2. This dissertation is the first to use conceptual clustering to partition coalition formation algorithms by mining crucial patterns and intricate relationships among the coalition formation algorithms. *i-CiFHaR* leverages the unsupervised learning method to cluster the algorithms in its library; thereby, analyzing only the most pertinent partition of algorithms for the application at hand. The use of clustering renders *i-CiFHaR* scalable, when new algorithms are added to its library the computation expense is not detrimentally impacted.

3. The primary contributions to the swarm intelligence community are the novel hybrid biologically inspired ACO algorithms that leverage the advantages of both the ant colony optimization and the simulated annealing techniques in order to solve combinatorial optimization problems. *sA-ANT* and *sA-ANT** employ two novel pheromone update policies that are radically different from conventional ACO approaches by integrating the simulated annealing technique. *sA-ANT* leverages a dynamically modulated number of ants to deposit pheromones, while *sA-ANT** maintains a dynamically modulated repository of solutions and uses the single best fit ant from the repository for pheromone depositing. Equipped with a high initial annealing temperature and a gradual annealing schedule, both algorithms provide enhanced search exploration during the initial phases, followed by improved exploitation

during the later phases, without stagnating in a local optima. The hybrid ant colony optimization algorithms effectively address the search stagnation shortcoming of contemporary ACO methods; thereby, generating significantly better results when applied to three *NP*-complete problems.

4. A distributed version of *sA-ANT** for the coalition formation problem is presented that contributes to the multi-agent and multi-robot systems communities. The algorithm computes high quality coalitions based on an information propagation mechanism, where every robot communicates with its immediate neighbors and publishes any newly acquired knowledge about the world. Over time, the robots capture as much global information as possible, given realistic real-world communication ranges. The decentralized *d-sA-ANT** algorithm is the first ACO-based coalition formation algorithm that has been simulated for the coalition problem in a real-world, multi-robot settings.

V.3 Future Work

A number of future research work can emerge from this dissertation. Some of the potential directions are described below.

V.3.1 Potential Extensions to *i-CiFHaR*

The current *i-CiFHaR* framework can be extended in a number of ways: (1) compute hybrid coalitions comprised of both humans and robots, (2) coalition formation algorithms can be added to its library, and (3) integration of planning.

V.3.1.1 Hybrid Coalition Formation

i-CiFHaR is the harbinger for future hybrid coalition formation for teams comprised of both humans and robots for critical missions (e.g., planetary, military, disaster response). Both a human and robot can constitute an agent in a multi-agent settings, where robots may have either non-consumable resources (e.g., camera, laser, sonar) or depletable resources (e.g., battery, fuel, ammunition), while humans have capabilities (e.g., fatigue, environmental factors, training) that can be modeled using *human performance moderator functions* (Silverman et al., 2006). *i-CiFHaR* can evolve to form hybrid coalitions by incorporating models of human agents' relevant performance factors. Some human performance factors (e.g., rank, height, training, and skills) do not change rapidly, while other performance factors (e.g., workload, fatigue, reaction time) can change quickly. Many human performance factors can be modeled and predicted (Harriott et al., 2013). Currently, tasks are specified by their resource or service requirements that can be accomplished by robots; however, future research can expand on the same idea with task descriptions including performance attributes, such as:

- **Workload** - Amount of physical work required by a task, given the duration of the task. Workload of a task can be decomposed into seven workload channels: (1) Auditory, (2) Visual, (3) Speech, (4) Tactile, (5) Cognitive, (6) Fine Motor, and (7) Gross Motor (Harriott et al., 2013).
- **Expected Fatigue Requirement** - Fatigue level, resulting from factors, such as sleeplessness, stress, etc. that will be elicited during the task execution and the values are assumed to be available from existing research.
- **Rank in the organization** - The rank of the human in the organization who is eligible to perform the task.

- **Training skills** - A set of training skills with respective levels that is required for task accomplishment.
- **Reaction/Response time** - Reaction time, which is the time required by an individual to respond to a particular stimulus, while performing a task. Reaction time is affected by several factors (e.g., age, gender, fatigue) and other human performance factors, such as brain injury, etc. (Harriott et al., 2013).
- **Environment characteristics** - Conditions characterizing the task environment in which the task will be conducted, such as temperature, humidity, wind speeds, light (day/night), etc.
- **Gear Requirement** - Defines the type of gear a human needs to wear while performing the task (e.g., Mask and Hood, Gloves, Overgarmet and Helmets).

Similar to the resource capabilities of robots, each human agent in the team will have attributes associated with each human performance factor that may contain:

- **Prior performance** - The prior performance or experience gained by a human for a particular type of mission.
- **Training** - The level of training a human has acquired for a particular type of mission.
- **Stress level** - The stress levels of a human that can originate from fatigue, sleeplessness, etc.
- **Rank within organization** - The rank to which the human belongs in an organization.
- **Visibility and Reaction time** - Internal performance factors of the human.
- **Workload capacity** - The workload reserve of the human that dictates how long the human can perform a task.
- **Gear level** - The level of personal protective gear that a human wears.

Once the additional human attributes are determined and the humans are appropriately represented, *i-CiFHaR* will consider both the human and robotic assets as abstracted agents for the coalition formation algorithms to compute the hybrid task coalitions.

V.3.1.2 Expanding *i-CiFHaR*'s library

A number of newly developed coalition formation algorithms can be added to *i-CiFHaR*'s library of algorithms. The developed *sA-ANT*, *sA-ANT**, and *d-sA-ANT** approaches tailored for the multi-agent coalition formation problem can be added to the library by classifying the algorithms in accordance with Service and Adams' taxonomy. Some other coalition formation algorithms (Ramchurn et al., 2010; Shiroma and Campos, 2009; Xu and Li, 2008) also need to be implemented for the framework. *i-CiFHaR*'s built-in conceptual clustering will accommodate these new algorithms into suitable clusters; thereby, either retaining the current hierarchical cluster tree of algorithms or creating a completely new hierarchical structure.

V.3.1.3 Integration of planning

Currently, *i-CiFHaR* is capable of selecting the most pertinent coalition formation algorithms to apply to a given mission, including the generation of agent/robot coalitions for the tasks. However, completion of the allocation task requires planning, and not just the coalition members. Planning needs to be effectively integrated into *i-CiFHaR*, such that appropriate plans can be generated to allow each coalition member to accomplish the assigned task.

V.3.2 Extensions to *d-sA-ANT**

The *d-sA-ANT** algorithm needs to be implemented on real robots in order to study the impact of real-world communications on the algorithm's performance. Currently, the robots compute the coalitions based on their static positions. During real-world applications, robots can continuously move when computing the coalitions; therefore, the communication topology will dynamically change over time as robots move in and out of each other's communication ranges. First, the presented evaluations need to be repeated in simulation to realize the effect of moving robots on the performance. The algorithm can then be implemented on real robot hardware in order to realize *d-sA-ANT**'s impact on multi-robot systems. It is hypothesized that the number of communication messages will be quite high, due to the intense information propagation mechanism of the algorithm.

BIBLIOGRAPHY

Abdallah, S. and Lesser, V. (2004). Organization-based cooperative coalition formation. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 162–168.

ACO-Software (2014). Ant colony optimization-public software. http://www.aco-metaheuristic.org/aco-code/public-software.html.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035.

Ashlock, D. (2006). *Evolutionary computation for modeling and optimization*, volume 200. Springer Science & Business Media.

Baluja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In *Proceedings of the Twelfth International Conference on Machine Learning,*, pages 38–46.

Bullnheimer, B., Hartl, R. F., and Strauss, C. (1997). A new rank based version of the ant system - A computational study. *Central European Journal for Operations Research and Economics*, 7:25–38.

Campbell, A., Wu, A. S., and Shumaker, R. (2008). Multi-agent task allocation: Learning when to say no. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pages 201–208.

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

Chaimowicz, L., Cowley, A., Gomez-Ibanez, D., Grocholsky, B., Hsieh, M., Hsu, H., Keller, J., Kumar, V., Swaminathan, R., and Taylor, C. (2005). Deploying air-ground multi-robot teams in urban environments. *Multi-Robot Systems. From Swarms to Intelligent Automata*, 3:223–234.

Chaimowicz, L., Cowley, A., Sabella, V., and Taylor, C. J. (2003). ROCI: A distributed framework for multi-robot perception and control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 266–271.

Chaimowicz, L., Sugar, T., Kumar, V., and Campos, M. F. (2001). An architecture for tightly coupled multi-robot cooperation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2992–2997.

Christensen, A. L., O'Grady, R., and Dorigo, M. (2009). From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation*, 13(4):754–766.

Cohen, M. D. and Axelrod, R. (1984). Coping with complexity: The adaptive value of changing utility. *The American Economic Review*, pages 30–42.

Cordon, O., de Viana, I. F., Herrera, F., and Moreno, L. (2000). A new ACO model integrating evolutionary computation concepts: The best-worst ant system. *Proceedings of ANTS 2000 - From Ant Colonies to Artificial Ants*, pages 22–29.

Cyert, R. and Degroot, M. (1979). Adaptive utility. In Allais, M. and Hagen, O., editors, *Expected utility hypotheses and the allais paradox*, volume 21 of *Theory and Decision Library*, pages 223–241. Springer Netherlands.

Dang, V. D. and Jennings, N. (2004a). Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the Third International Joint Conference on Autonomous Agents and MultiAgent Systems*, pages 564–571.

Dang, V. D. and Jennings, N. R. (2004b). Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 564–571.

DeJong, P. (2005). Coalition formation in multi-agent UAV systems. MS Thesis, University of Central Florida.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

Dias, M. B. (2004). *TraderBots: A new paradigm for robust and efficient multirobot coordination in dynamic environments*. PhD Thesis, Carnegie Mellon University.

Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39.

Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(1):29–41.

Dudek, G., Jenkin, M. R., Milios, E., and Wilkes, D. (1996). A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397.

Dunteman, G. H. (1989). *Principal components analysis*. Sage Publications.

Durbin, R. and Willshaw, D. (1987). An analogue approach to the travelling salesman problem using an elastic net method. *Nature*, 326(6114):689–691.

Farinelli, A., Iocchi, L., and Nardi, D. (2004). Multirobot systems: A classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(5):2015–2028.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2(2):139–172.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communication of the ACM*, 5(6):345–349.

Freedy, A., Sert, O., Freedy, E., McDonough, J., Weltman, G., Tambe, M., Gupta, T., Grayson, W., and Cabrera, P. (2008). Multiagent adjustable autonomy framework (MAAF) for multi-robot, multi-human teams. In *Proceedings of the International Symposium on Collaborative Technologies and Systems*, pages 498–505.

Gambardella, L. M., Dorigo, M., et al. (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 252–260.

Gaston, M. E. and desJardins, M. (2005). Agent organized networks for dynamic team formation. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 230–237.

Gerkey, B. and Matarić, M. J. (2002). Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768.

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

Gluck, M. (1985). Information, uncertainty and the utility of categories. In *Proceedings of the Seventh Annual Conference on Cognitive Science Society*, pages 283–287.

Grefenstette, J., Gopal, R., Rosmaita, B., and Van Gucht, D. (1985). Genetic algorithms for the traveling salesman problem. In *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 160–168.

Grne-Yanoff, T. and Hansson, S. (2009). Preference change: An introduction. In Grne-Yanoff, T. and Hansson, S., editors, *Preference Change*, volume 42 of *Theory and Decision Library*, pages 1–26. Springer Netherlands.

Haque, M. A. and Egerstedt, M. (2009). Coalition formation in multi-agent systems based on bottlenose dolphin alliances. In *Proceedings of the American Control Conference*, pages 3280–3285.

Harriott, C. E., Zhang, T., and Adams, J. A. (2013). Assessing physical workload for human-robot peer-based teams. *International Journal of Human-Computer Studies*, 71(7):821–837.

Ilie, S. and Bădică, C. (2013). Multi-agent approach to distributed ant colony optimization. *Science of Computer Programming*, 78(6):762–774.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323.

Jolliffe, I. (1972). Discarding variables in a principal component analysis. I: Artificial data. *Journal of the Royal Statistics Society. Series C (Applied Statistics)*, 21(2):160–173.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kjærulff, U. B. and Madsen, A. L. (2008). *Bayesian networks and influence diagrams: A Guide to construction and analysis*. Springer Publishing Company.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

Koes, M., Nourbakhsh, I., and Sycara, K. (2005). Heterogeneous multirobot coordination with spatial and temporal constraints. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 1292–1297.

Lagoudakis, M. G., Berhault, M., Koenig, S., Keskinocak, P., and Kleywegt, A. J. (2004). Simple auctions with performance guarantees for multi-robot task allocation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 698–705.

Lau, H. C. and Zhang, L. (2003). Task allocation via multi-agent coalition formation: Taxonomy, Algorithms and Complexity. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 346–350.

Lemaire, T., Alami, R., and Lacroix, S. (2004). A distributed task allocation scheme in multi-UAV context. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 3622–3627.

Li, C. and Biswas, G. (2002). Unsupervised learning with mixed numeric and nominal data. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):673–690.

Liu, Y. and Nejat, G. (2013). Robotic urban search and rescue: A survey from the control perspective. *Journal of Intelligent and Robotic Systems*, 72(2):147–165.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297.

Martens, D., De Backer, M., Haesen, R., Vanthienen, J., Snoeck, M., and Baesens, B. (2007). Classification with ant colony optimization. *IEEE Transactions on Evolutionary Computation*, 11(5):651–665.

Mascia, F. (2014). Dimacs benchmark set. http://iridia.ulb.ac.be/~fmascia/maximum_clique/.

McLellan, C. and Harpstead, E. (2014). Concept formation. https://github.com/cmaclell/concept_formation.git.

Merkle, D., Middendorf, M., and Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6(4):333–346.

Middendorf, M., Reischle, F., and Schmeck, H. (2000). Information exchange in multi colony ant algorithms. In Rolim, J., editor, *Parallel and Distributed Processing*, volume 1800 of *Lecture Notes in Computer Science*, pages 645–652. Springer Berlin Heidelberg.

Neumann, L. J. and Morgenstern, O. (1947). *Theory of games and economic behavior*, volume 60. Princeton University Press Princeton, NJ.

Nielsen, T. D. and Jensen, F. V. (2004). Learning a decision maker's utility function from (possibly) inconsistent behavior. *Artificial Intelligence*, 160(1):53–78.

Nokia (2012). Qt. http://www.qt.io/.

NORSYS (2012). Netica application: A complete software package to solve problems using bayesian belief networks and influence diagrams. http://www.norsys.com/netica.html.

Oliveira, S. M., Hussin, M. S., Stützle, T., Roli, A., and Dorigo, M. (2011). A detailed analysis of the population-based ant colony optimization algorithm for the TSP and the QAP. In *Proceedings of the 13th annual conference companion on Genetic and Evolutionary Computation*, pages 13–14.

Page, L., Sergey, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University.

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

Pedemonte, M., Nesmachnow, S., and Cancela, H. (2011). A survey on parallel ant colony optimization. *Applied Soft Computing*, 11(8):5181–5197.

Pedersen, L., Kortenkamp, D., Wettergreen, D., and Nourbakhsh, I. (2003). A survey of space robotics. In *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, pages 19–23.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell system technical journal*, 36(6):1389–1401.

Rahwan, T. (2007). *Algorithms for coalition formation in multi-agent systems*. PhD thesis, University of Southampton.

Rahwan, T. and Jennings, N. R. (2008a). An improved dynamic programming algorithm for coalition structure generation. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1417–1420.

Rahwan, T. and Jennings, N. R. (2008b). Coalition structure generation: Dynamic programming meets anytime optimisation. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*, pages 156–161.

Rahwan, T., Ramchurn, S., Jennings, N., and Giovannucci, A. (2009a). An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, 34:521–567.

Rahwan, T., Ramchurn, S. D., Jennings, N. R., and Giovannucci, A. (2009b). An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, 34(2):521–567.

Ramchurn, S. D., Polukarov, M., Farinelli, A., Truong, C., and Jennings, N. R. (2010). Coalition formation with spatial and temporal constraints. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1181–1188.

Randall, M. and Lewis, A. (2002). A parallel implementation of ant colony optimization. *Journal of Parallel and Distributed Computing*, 62(9):1421–1432.

Reinelt, G. (2014). TSPLIB. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

Ren, Z., Feng, Z., and Wang, X. (2008). An efficient ant colony optimization approach to agent coalition formation problem. In *Proceedings of the 7th World Congress on Intelligent Control and Automation*, pages 7879–7882.

Rothkopf, M. H., Pekeč, A., and Harstad, R. M. (1998). Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147.

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

Sariel-Talay, S., Balch, T., and Erdogan, N. (2011). A generic framework for distributed multirobot cooperation. *Journal of Intelligent and Robotic Systems*, 63(2):323–358.

Sen, S. and Adams, J. (2013a). sA-ANT: A hybrid optimization algorithm for multirobot coalition formation. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 2, pages 337–344.

Sen, S. and Adams, J. (2014). An influence diagram based multi-criteria decision making framework for multirobot coalition formation. *Autonomous Agents and Multi-Agent Systems*, pages 1–30.

Sen, S. D. and Adams, J. A. (2013b). A decision network based framework for multiagent coalition formation. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, pages 55–62.

Service, T. C. (2009). Theory and algorithms for coalition formation among heterogeneous agents. Technical Report Rep. HMT-09-01, Vanderbilt University.

Service, T. C. and Adams, J. A. (2010). Coalition formation algorithm taxonomy. Technical Report HMT-10-03, Vanderbilt University.

Service, T. C. and Adams, J. A. (2011a). Coalition formation for task allocation: Theory and algorithms. *Journal of Autonomous Agents and Multi-Agent Systems*, 22(2):225–248.

Service, T. C. and Adams, J. A. (2011b). Constant factor approximation algorithms for coalition structure generation. *Autonomous Agents and Multi-Agent Systems*, 23(1):1–17.

Service, T. C., Sen, S. D., and Adams, J. A. (2014). A simultanesous descending auction for multirobot task allocation. In *Proceedings of the 2014 IEEE International Conference on Systems, Man, and Cybernetics*.

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

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

Silverman, B. G., Johns, M., Cornwell, J., and O'Brien, K. (2006). Human behavior models for agents in simulators and games: Part I: Enabling science with PMFserv. *Presence Teleoperators and Virtual Environments*, 15(2):139–162.

Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 100(12):1104–1113.

Solnon, C. and Fenet, S. (2006). A study of ACO capabilities for solving the maximum clique problem. *Journal of Heuristics*, 12(3):155–180.

Song, F., Guo, Z., and D.Mei (2010). Feature selection using principal component analysis. In *Proceedings of the International Conference on System Science, Engineering Design and Manufacturing Informatization*, pages 27–30.

Sparrow, R. (2009). Building a better warbot: Ethical issues in the design of unmanned systems for military applications. *Science and Engineering Ethics*, 15(2):169–187.

Stutzle, T. and Hoos, H. H. (2000). MAX-MIN ant system. *Future Generation Computer Systems*, 16(8):889–914.

Sujit, P. B., George, J. M., and Beard, R. W. (2008). Multiple UAV coalition formation. In *Proceedings of the American Control Conference*, pages 2010–2015.

Talavera, L. and Béjar, J. (2001). Generality-based conceptual clustering with probabilistic concepts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):196–206.

Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7(1):83–124.

Tambe, M., Pynadath, D. V., Chauvat, N., Das, A., and Kaminka, G. A. (2000). Adaptive agent integration architectures for heterogeneous team members. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, pages 301–308.

Tang, F. and Parker, L. E. (2005). ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1501–1508.

Tews, A. and Wyeth, G. (1998). Using centralised control and potential fields for multi-robot cooperation in robotic soccer. In *Proceedings of Pacific Rim International Workshop on Multi-Agents*, pages 22–27.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. The MIT Press, 1st edition.

Tošić, P. T. and Agha, G. A. (2005). Maximal clique based distributed coalition formation for task allocation in large-scale multi-agent systems. In *Proceedings of the Massively Multi-Agent Systems I*, pages 104–120.

Twomey, C., Stützle, T., Dorigo, M., Manfrin, M., and Birattari, M. (2010). An analysis of communication policies for homogeneous multi-colony aco algorithms. *Information Sciences*, 180(12):2390–2404.

Vig, L. and Adams, J. A. (2006a). Market-based multi-robot coalition formation. In *Proceedings of the 8th International Symposium on Distributed Autonomous Robotic Systems*, pages 227–236.

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

Weerdt, M., Zhang, Y., and Klos, T. (2007). Distributed task allocation in social networks. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 17–24.

Xia, N., Jiang, J., and Hu, Y. (2004). Solution to agent coalition problem using improved ant colony optimization algorithm. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 475–478.

Xia, Y. and Xi, B. (2007). Conceptual clustering categorical data with uncertainty. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, volume 1, pages 329–336.

Xie, L. and Mei, H. (2007). The application of the ant colony decision rule algorithm on distributed data mining. *Communications of the International Information Management Association*, 7(4):85–94.

Xu, J. and Li, W. (2008). Solution of overlapping coalition formation based on discrete particle swarm optimization. In *Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4.

Yu, Q. and Terzopoulos, D. (2007). A decision network framework for the behavioral animation of virtual humans. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 119–128.

Zhang, G., Jiang, J., Su, Z., Qi, M., and Fang, H. (2010). Searching for overlapping coalitions in multiple virtual organizations. *Information Sciences*, 180(17):3140–3156.

Zhang, Y. and Parker, L. E. (2010). IQ-ASyMTRe: Synthesizing coalition formation and execution for tightly-coupled multirobot tasks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5595–5602.