TOKEN-BASED APPROACH FOR SCALABLE TEAM COORDINATION

by

Yang Xu

PhD of Information Sciences

Submitted to the Graduate Faculty of

in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2007

UNIVERSITY OF PITTSBURGH

This dissertation was presented

by

Yang Xu

PhD of Information Sciences

It was defended on

December 12, 2007

and approved by

ACKNOWLEDGMENTS

First I would like to acknowledge my dissertation committee members: Dr. Michael Lewis, Dr. Paul Scerri, Dr. Katia Sycara, Dr. Paul Munro and Dr. Marek Druzdzel. Dr. Lewis and Dr. Scerri, thanks for being my advisors, and for giving me great advice along the way. This truly made me a better investigator, a quality that will undoubtedly be beneficial in my future career. Paul, thanks for helping me out with every aspect of my project and for being not only an exceptional teacher, but also a friend. It was my pleasure to work closely with you for past years. Dr. Sycara, your suggestions and comments really helped me to think critically. I have always appreciated your great advice and many words of wisdom. Dr. Munro and Dr. Druzdzel, thanks for all of your help and suggestions for my dissertation project. I learned a lot from all of you, and my committee really played an enormous role in the success of my dissertation project.

I must also thank three people who helped me out so much. Bin Yu, you were always able to make me feel more at ease and comfortable. Thanks for having been so willing to help me with everything. Jijun and Guoming, my partners working in the same lab, I will always remember the time we struggled and laughed together.

There are so many people in intelligent software agent lab to thank also: Joseph Giampapa, Robin Glinton and Sean Owen. I appreciate all the time and effort you put into helping me.

I would like to thank the students in School of Information Science. In particular, I would like to thank Ming Mao, who is so warm-hearted and helped my family a lot.

I would also like to thank my family for all of their support over the years. My mother, father and sister, thank you for always being there for me and encouraging me to go as far as I can in life. Thanks for always being proud of me no matter what. Specifically, I would also like to thank my mother- and father-in-law for supporting me and taking good care of my wife and my baby. And, last but not least, I must thank my wife, Li. Thank you for always believing in me, being there for me and taking good care of our family. And, finally, my future daughter Tiantian, for putting things in perspective and giving me more power to get though any situation.

TOKEN-BASED APPROACH FOR SCALABLE TEAM COORDINATION

Yang Xu

PhD of Information Sciences, PhD

University of Pittsburgh, 2007

To form a cooperative multiagent team, autonomous agents are required to harmonize activities and make the best use of exclusive resources to achieve their common goal. In addition, to handle uncertainty and quickly respond to external environmental events, they should share knowledge and sensor in formation. Unlike small team coordination, agents in scalable team must limit the amount of their communications while maximizing team performance. Communication decisions are critical to scalable-team coordination because agents should target their communications, but these decisions cannot be supported by a precise model or by complete team knowledge.

The hypothesis of my thesis is: local routing of tokens encapsulating discrete elements of control, based only on decentralized local probability decision models, will lead to efficient scalable coordination with several hundreds of agents. In my research, coordination controls including all domain knowledge, tasks and exclusive resources are encapsulated into tokens. By passing tokens around, agents transfer team controls encapsulated in the tokens. The team benefits when a token is passed to an agent who can make use of it, but communications incur costs. Hence, no single agent has sole responsible over any shared decision. The key problem lies in how agents make the correct decisions to target communications and pass tokens so that they will potentially benefit the team most when considering communication costs.

My research on token-based coordination algorithm starts from the investigation of random walk of token movement. I found a little increase of the probabilities that agents make the right decision to pass a token, the overall efficiency of the token movement could be greatly enhanced. Moreover, if token movements are modeled as a Markov chain, I found that the efficiency of passing tokens could be significantly varied based on different network topologies.

My token-based algorithm starts at the investigation of each single decision theoretic agents. Although under the uncertainties that exist in large multiagent teams, agents cannot act optimal, it is still feasible to build a probability model for each agents to rationally pass tokens. Specifically, this decision only allow agent to pass tokens over an associate network where only a few of team members are considered as token receiver.

My proposed algorithm will build each agent's individual decision model based on all of its previously received tokens. This model will not require the complete knowledge of the team. The key idea is that I will make use of the domain relationships between pairs of coordination controls. Previously received tokens will help the receiver to infer whether the sender could benefit the team if a related token is received. Therefore, each token is used to improve the routing of other tokens, leading to a dramatic performance improvement when more tokens are added. By exploring the relationships between different types of coordination controls, an integrated coordination algorithm will be built, and an improvement of one aspect of coordination will enhance the performance of the others.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION

With the development of intelligent agent research, single agents are required to cooperative with others and formed as a team when they cannot solve problems themselves or achieve goals out their own. In such a cooperative multiagent team, unselfish autonomous agents should jointly make decisions to harmonize their activities and make the best use of exclusive resources accomplish their common goal. Moreover, to handle uncertainty and quickly respond to external environmental events, team members should share their knowledge and sensor information, and be capable of adjusting their activities automatically when those activities cannot produce the expected results.

Coordination within those teams are to enable agents to cooperate more efficiently when pursuing their goals. Existing coordination algorithms have been developed including market-based approaches [26], teamwork approaches [91], decision theoretical approaches [11], etc. Their applications have been successfully implemented in teams supporting human collaboration[18, 87], teams for disaster response[60], for manufacturing[44], for training[92] and for games[45]. However, state-of-the-art coordination applications require hundreds or thousands of agents or robots working together. Their goal could be extremely complex, distributed and their open environment is highly dynamic, emergent, ad-hoc, or even hostile. For example, in a large scale disaster response, hundreds of fire fighters, paramedics, and many others need to work together, albeit loosely, to mitigate the effects of the disaster. Those applications are critical in real-time coordination domains especially in military, astronomy and disaster response, such as mobile sensors for target tracking[48]; unmanned aerial vehicles for battlefield[25]; and spacecraft teams for planet explorations[65]. Different with the small-team coordination applications, agents in those domains are highly distributed with limited communication capabilities, but they have to independently make their decisions.

1.1 CHALLENGES

To address those large-team coordination applications above, we need the coordination algorithms to be capable of handling substantially big teams but to retain the comparable efficiency of coordinating a small team. Unfortunately, existing team coordination algorithms [89, 47, 26] are not capable of coordinating agents when their team size scales up. The major challenges include communication limitation, decentralized control, and incomplete knowledge for decision support. In the rest of this section, I describe those challenges in more details.

1.1.1 Limitations in communication

Communication limitation is the primary challenge when we apply existing team coordination algorithms to a large team. Existing algorithms for small-team coordination assume that communication is not under constraints, and agents can freely communicate with any other agents. Under this assumption, in existing coordination approaches, centralized communication protocols, such as a blackboard [15] or specific centralized agents, such as information agents [33], are designed.

However, communication in a large team is constrained for two reasons. First, with a physically finite communication bandwidth, agents in a large team cannot communicate with all their teammates when the team size is hundreds of times larger than a small team. Second, agents in a large team normally more widely distributed than in a small team. Their physical bandwidth is less than when they are in a small team, either because of a weaker wireless signal or because more transmissions are required to deliver a message. In this viewpoint, communication in a large multiagent team has been a valuable resource. Excessive communication protocols are either infeasible, undesirable, or too expensive. To reduce communication, coordination either adopts a silent stratagem [105] or attempts to

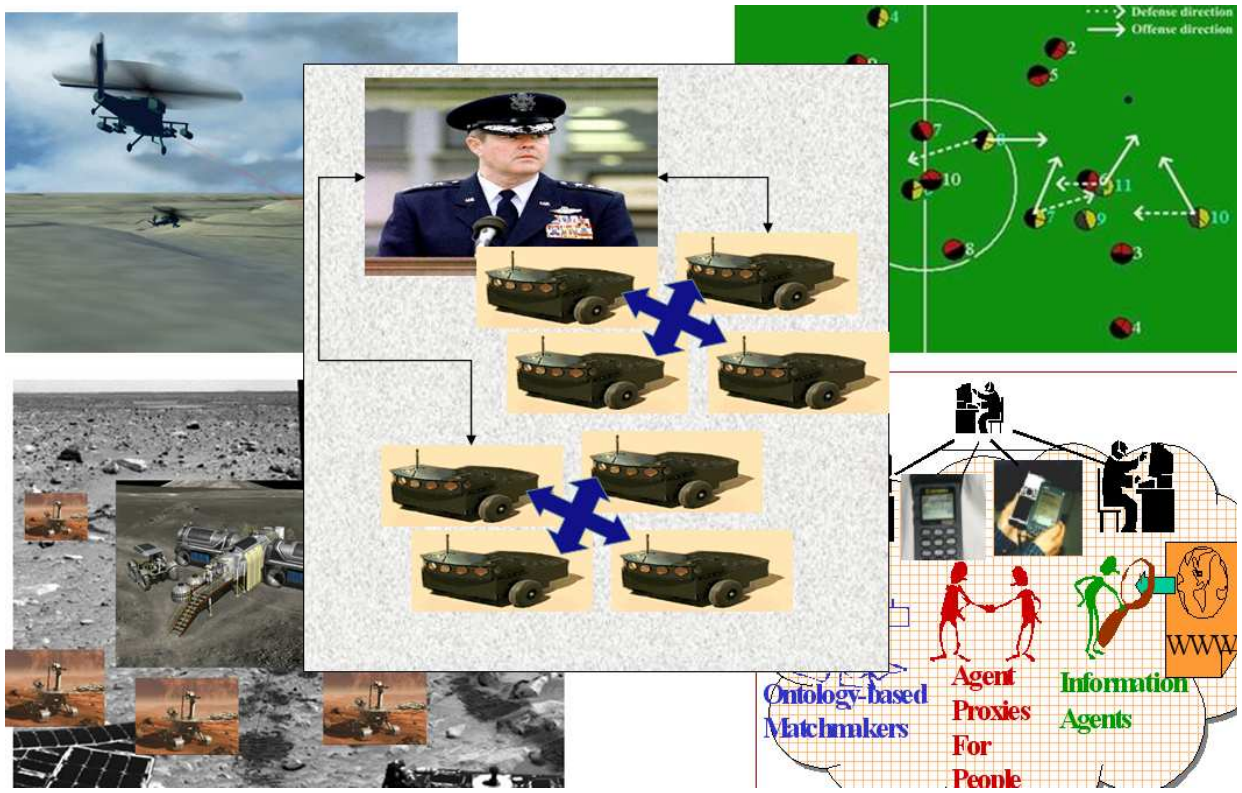


Figure 1: State-of-art coordination applications require hundreds or thousands of agents or robots working together towards complex goals. (a) Coordination unmanned helicopters (b)RoboCup (c) Exploration of Mars (d) Mobile agents over internet (e) Coordination of military vehicles

target agents' communication [102] to only one or a small number of teammates who can potentially contribute to the team after the message is received. Unfortunately, both methods in existing coordination approaches require the decision support based on a precise decision model or enough knowledge of the team states. These requirements will be explained in the following sections to be hard for a large team.

1.1.2 Decentralized control

Although in both small-team coordination and scalable-team coordination, agents are distributed in an open environment, agents in small teams are assumed to be able to observe the complete team state or gain them via broadcast or other centralized communication protocols, such as a blackboard [15] or an information agent [33]. By designing centralized algorithms, such as auction [26], decision theoretical agent [12] or GPGP/TAEMS [47], small-team coordination tried to find the optimal joint policies for team members to act in their environments in the center units.

Unlike small-team coordination, scalable-team coordination requires decentralized controls. In scalable-team coordination, although agents work together to achieve their common goal, they must independently make decisions to their own activities. There are three main reasons for this. First, due to the restrictions on communication mentioned above, no agent is capable of acting as the central unit, such as a supervisor or auctioneer that is able to listen and dispatch orders to all the other team members. Second, the computations required to search an efficient joint policy for a large team in a single unit is either practically too expensive or impossible. Third, centralized-control applications are vulnerable in certain domains, such as the military. The entire team will halt even if the only agent acts as central unit fails.

1.1.3 Incomplete team states and environment

In scalable team coordination algorithms, distributed agents are always only able to sense a part of the team state and their environment. In some of the ad-hoc environments, they may not even know how many teammates there are. For example, an Unmanned Aerial Vehicle

(UAV) that is involved in a military operation may observe many features of the battlefield on the route to an assignment. Many of its observations will be relevant to the assignments of other combatants, but the UAV may not necessarily know which members require those information. Events that are relevant to team goals will become available to some team members in a spontaneous, unpredictable, and, most importantly, distributed way. To be able to make rational decisions under uncertainty of the environment, communication is required so that agents can obtain sufficient knowledge of the team state and react to those dynamic environmental changes.

On the other hand, in existing team coordination models, team members are assumed to have precise models of the team state to support their decisions in order to achieve their joint goals [43, 92]. However, when team size is scaled up, it becomes unfeasible to maintain up-to-date, detailed models of all other teammates and team activities, because of agents' limited communication abilities and sensor perceptions. Without these models, key elements of both the theory and operationalization of existing teamwork break down. For example, without accurate models of team activities, STEAM's communication reasoning [92] cannot be applied, nor can Joint Intention's reasoning about commitments [43]. Therefore, a key challenge for scalable team coordination is how to design a rational decision mechanism that is based on an imprecise model of the team. It has been a chicken-and-egg puzzle, where rich localized knowledge is indispensable to target communications, while communication is required for sharing team knowledge.

Furthermore, when the size of a team is scaled up, more variables are included in the team state. Team state space has been exponentially expanded according to the number of variables. Hence, to be computationally feasible to search a rational decision, the team state has to be factored and only the variables that potentially contribute to current decision should be taken into consideration. Another challenge exists because without sufficient knowledge of the team states, agents do not know which variables should be taken into account.

1.1.4 Heterogeneous Team

Coordinating a heterogeneous team [68, 90] is another challenge when the members of a large team have different capabilities for different roles. When coordinating such a heterogeneous team, there are more constraints on role and resource allocation. In these teams, a specific role can only be taken by some agents who are capable of that role, and different quantities of team reward (a measure of team utility function for "good performance") will be expected when it is performed by different capable agents. Moreover, specific exclusive resources are required for agents to take a role. For example, only the robots with medical training are capable of victim rescue, and to fight of a fire, a resource of hydrant is required. Unlike the tight constraints in small robotics team coordination, role assignments can be performed independently in scalable coordination. The major challenge is that no agent has the complete knowledge of the capabilities of their teammates. Therefore, in scalable coordination, without enough knowledge of the team state, agents are hard to make decisions of how to assign a role or a resource to a teammate in a way that will benefit the team most.

1.1.5 Solving Conflicts

With the increasing of complexity, dynamics, and uncertainty in scalable teamwork, agents' coherent joint activities are inevitably jeopardized by conflicts in agents' beliefs, plans, and actions [49, 98]. Based on the incomplete knowledge of team and the environment, agents cannot always make correct decisions. Conflicts occur in several scenarios. For examples, a noisy reading from a sensor may create an incorrect piece of information; a fast-changing environment puts agents' inferences of the team state out of date very quickly; an agent duplicates a plan without noticing that other agents have created the same plan; or one or more agents perform the same activity without noticing that the activity has been duplicated. The successful handling of potential conflicts within a scalable team is required before a robust coordination approach is carried out.

1.1.6 Social Effects

When teams scale up, social organization topology is an important factor to be considered in coordination. The efficiency of a large team is not simply the sum of all efforts of all team members. Previous research has revealed that social structures, patterns, and interconnections have a strong impact on the effectiveness of communication and cooperation within both human society [58, 100] and artificial computation-based multiagent teams [55]. The most popular manifestation of this phenomenon is the *six degrees of separation* concept, which was uncovered by Milgram [58]. My research of " information sharing among scalable teams" has revealed that sharing efficiencies are significantly varied among different social network topologies [102]. Therefore, in a scalable team coordination design, it is important to include the successful social effects that fosters efficient behavior in team organization, which has not been considered in small-team coordination.

1.2 DISTRIBUTED COORDINATION APPROACHES

As explained in the previous section, small-team coordination approaches when applied in scalable-team coordination, the major issue is its incapability of making decisions under incomplete team knowledge with a limited communication. Initial research for scalable coordination has been made in multiagent communities, but those distributed multiagent coordination approaches have failed to produce algorithms or implementations that meet all the challenges. Most algorithms that have been developed to solve key problems do not scale to very large teams. For example, optimal search techniques in trading of computational complexity cannot be applied when the team state space is exponentially expanded [59, 61].

The rare algorithms that do scale effectively typically either make unreasonable assumptions or are very specialized to a specific problem (such as emergent behavior [75]). Coordination that relies on swarm-like behavior that, while robust, can be very inefficient [19]. Other approaches that have been shown to be scalable often rely on some degree of centralization, for example, using an auctioneer [41, 53, 34]. Those approaches may not always be desirable or feasible. Decentralized coordination models such as Decentralized Partial Observable Markov Decision Process (DEC-POMDP). Although they can mathematically model the problem well, their solutions are computationally NP-complete [9]. Therefore, new approaches are needed to be designed for scalable coordination.

1.3 INTEGRATED COORDINATION ALGORITHM

Autonomous coordination is a complex process since several distributed algorithms are required to interact to produce agile, cohesive, and efficient coordinated behavior. Typical teamwork requires different aspects of coordination, such as sharing information on external environmental events [102], allocating individual tasks for joint actions [29], initiating plans [104], detecting conflicts on knowledge and activities around the team [49], human interaction[84], and sharing exclusive resources[54].

There have been several existing coordination algorithms capable of distributed coordination, but they only focus on a partial aspect of coordination and overlook the interrelationship between them. For example, role allocation or planning algorithms in [80, 41] preclude the use of knowledge from other aspects of coordination that may improve the performance of their algorithms. Results of the role allocation process are not typically used to guide resource allocations, although intuitively they will improve the decisions. In information sharing algorithm [102], if an agent knows the tasks that other agents are carrying out, it can route the related information more efficiently. On the other hand, when agents know the status of one another, it is much easier to assign related tasks. Unfortunately, no existing research has shown that interrelationships between different aspects of coordination are being used to improve their algorithms.

A key innovation in my thesis is based on my observation that different coordination algorithms are interrelated. The relationships underlying different aspects of coordination tasks imply that an integrated decision model can be built, based on all of the coordination tasks, to more efficiently solve the coordination decision problem.

1.4 RESEARCH OBJECTIVE

Although initial efforts have been made in previous research to design scalable coordination algorithms, these efforts have not yet been able to accommodate all the challenges. The hypothesis of my thesis is that local routing of tokens encapsulating discrete elements of control, based only on decentralized local probability decision models will lead to efficient scalable coordination within teams comprised of several hundreds of agents. *The major task is to design an algorithm for distributed agents so that they can locally make rational decisions based on their limited view and incomplete knowledge to maximize team utility.* In my research, coordination controls, including all domain knowledge, tasks, and exclusive resources are encapsulated into tokens. By utilizing their limited communication bandwidth, agents target their communication and pass the tokens around to transfer team controls that are encapsulated in the tokens. The team is benefited when a token is passed to an agent who can make use of it, but communications incur costs. In the process of passing tokens, efficient coordination of task assignment, exclusive resources allocation, and sharing knowledge could be realized, but no single agents has sole responsibility for any shared decision.

The key problem is how agents make correct decisions based on their partial observations and team knowledge. My proposed algorithm will build each agent's local decision model based on all its previous incoming tokens. This model will not require the complete knowledge of the team. The key idea is that I will make use of the domain relevance between pairs of coordination controls. Previously received tokens help the receiver to infer whether the sender will benefit the team if a related token is received. Therefore, each token is used to improve the routing of other tokens leading to a dramatic performance improvement when the algorithms work together. By exploring the relationships between different types of coordination controls, an integrated coordination algorithm will be built, and one aspect of coordination will enhance the performance of the others.

My initial experiment (described in section 5.2) shows that a little enhancement of precision in local decision policy will greatly improve the performance of team coordination and the local communication decision process. Moreover, I will make use of the relationships between various coordination tasks and build an integrated decision model that is capable of performing every aspect of coordination tasks. In my research, I am mainly interested in three major aspects of coordination: information sharing, task assignment and exclusive resource allocation. The major advantage of this integrated model is that an improvement in one aspect of coordination, such as information sharing, will improve the performance of other aspects of coordination, such as role assignment and resource allocation.

In addition to the token-based coordination algorithm, a logical, static network across the team limits agents to forwarding tokens to their neighbors in this network. As a result, an agent directly receives tokens from a small number of neighbors in the network and can thus build much better models of those agents. In my thesis, I will conduct an investigation of whether teams that are organized according to social effects [6] enhance the team performance. I am interested in four major team organization topologies including random, grid, small world and scale free network. Specifically, small world and scale free effect have been verified as efficient for information sharing around large scale teams [102].

2.0 PROBLEM DESCRIPTION

In this section, I will formally describe the scalable-team coordination problem based on team oriented plan model. In this model, high-level common goal is decomposed into subgoals. In my research, I focus on three main coordination algorithms: information sharing, role assignment, and resource allocation. In this section, My formal model for scalable-team coordination is described. Then, a detailed example of scalable-team coordination domain is presented.

2.1 TEAM ORIENTED PLANS

Team Oriented Plans (TOPs) are the abstraction that define team behavior. The TOPs provide the mapping from team level goals to individual *roles* that are performed by individual team members. Suppose the team A has a top level goal, G. The team *commits*, with the semantics of STEAM to G [92]. Achieving G requires achieving sub-goals, g_i, that are not known in advance but are functions of the environment. For example, sub-goals of a high level goal to respond to a disaster could be to extinguish a fire and provide medical attention to particular injured civilians. To achieve sub-goals, the team follows plan templates represented in a library. These templates are parameterized while instantiated plans contain the specific details [69]. For example, when a particular fire in a building is detected by a team member, the plan will be instantiated because it matches a template for disaster response.

Individual agents may commit the team to a sub-goal, provided that it matches a plan template. Each sub-goal is addressed with a plan, $plan_i =< g_i, recipe_i, roles_i, events_i >$, that matches a plan template in the library. The overall team thus has plans $Plans(t) =$

$\{plan_1, \ldots, plan_n\}$. Individual team members will not necessarily know all plans. To maximize the responsiveness of the team to changes in the environment, we allow any team member to commit the team to executing a plan, when it detects that subgoal g_i is relevant. Team members can determine which sub-goals are relevant by the plan templates specified by the library. $Recipe_i$ is a description of the way the sub-goal will be achieved[43] including the execution order of the components in the plan. $Roles_i = \{role_1, role_2, role_3, ...role_r\}$ are the individual activities that must be performed to execute $recipe_i$. $event_i$ is the domain specific information pertinent to the plan. For convenience, we write $perform(r, a)$ to signify that agent, a, is working on role, r.

One way to think about TOPs is as active objects in a distributed database. Each TOP "object" captures the state of a particular team plan. Team members involved in the execution of that plan need to have up-to-date versions of the TOP "object", e.g., knowing which team members are performing which roles and when TOPs are complete. Information needs to be shared to ensure there is synchronization across the same object held by different team members. Viewed in this manner, coordination can be thought of as a set of algorithms to fill in fields on the TOP objects and ensure synchronized objects across the team. For example, some coordination algorithms are triggered when there are open roles in the TOP objects and other algorithms are triggered when the post-conditions on the plan are satisfied.

2.2 SCALABLE-TEAM COORDINATION MODEL

The team coordination problem is defined as: there is a scalable multiagent team $A = \{a_1, a_2, \ldots, a_{|A|}\}$ and a_i represents a specific agent i. They share a top level common goal G.

2.2.1 Teamwork

Based on the joint intentions framework [89] and team oriented plan model explained above, G can be realized by achieving a set of joint intention sub-goals $\{g_1, g_2, ..., g_i, ...\}$ in multiple plans. Each of the joint intention plan j for a sub-goal g_i is written as a tuple $plan_{i,j} =<$

$g_i, p_{i,j}, q_{i,j}, reward_i >$, where

- g_i is the subgoal.
- $p_{i,j} = \{event^1_{i,j}, event^2_{i,j}, ...\}$ are the preconditions to activate a specific plan j that can achieve subgoal g_i. Please note that we defined g_i can be achieved in several ways. For example, distinction a fire can achieved either by spraying water or fire distinct chemicals.
- $q_{i,j} = \{role^1_{i,j}, role^2_{i,j}, ...\}$ denotes all individual activities required to be performed by single agents after $plan_{i,j}$ is activated to realize g_i. Each $role_{i,j} =< task_{i,j}, ability_{i,j},$ $resource_{i,j} >$ is represented by its task, i.e., a description of the actual thing to be done; the capabilities required to perform that task and the resources needed to fill the role.
- $reward_i$ will be credited to team A when the sub-goal g_i is achieved.

For example, a plan of fire fighting can be defined as: <(Fire fighting at a location X), (Notice of a fire alarm at X $\cap$ Observation of smoke at X), $(role^1_{i,j}, role^3_{i,j}, role^3_{i,j})$, (100)>. This template requires two conditions before it is initiated: notice of a fire alarm and observation of smoke. After this plan is initiated, three roles may need to be allocated. $role^1_{i,j}$=<(Driving the fire truck), (Skillful in driving truck), (An available fire truck)>, $role^2_{i,j}$=<(fighting the fire), (Have training in fire fighting), (Fire fighting equipment)> and $role^3_{i,j}$=<(Searching for victims), (None), (Fire fighting equipment)> are three roles in this template: driving the fire truck, fighting the fire and searching for victims. To perform $role^1_{i,j}$, an agent is required to be able to drive and have access to a fire truck (resource). After all the roles are located, a reward 100 will be credited to the team.

I define $INF = \{event^1_{1,1}, event^2_{1,1}, ..., event^k_{i,j}, ...\}$ as the set of all possible domain events; $ROLE = \{role^1_{1,1}, role^2_{1,1}, ..., role^k_{i,j}, ...\}$ is the set of potential available activities and $RESOURCE = \{res_1, res_2, ...res_k, ...\}$ is all available exclusive resources in team A.

The capability of agent a_i to perform $role^k_{i,j}$ is mapped as a quantitative value given by: $Cap(a_i, role^k_{i,j}) \rightarrow [0, 1]$, e.g., $Cap(a_i, (Drivingfiretrunk)) = 0.8$. I also write the resource requirements for a_i to perform $role^k_{i,j}$ as $RequireRes(a_i, role^k_{i,j}) \subseteq RESOURCE$ (e.g., $RequireRes(a_i, (Drivingfiretrunk)) = \{firetrunk\}$) and the resources that are currently available for a_i as $AvailableRes(a_i) \subseteq RESOURCE$ (e.g., $AvailableRes(a_i) = \{hydrant\}$). Whether agent a_i is able to perform $role^k_{i,j}$ depends on its capability and available resources.

Formally, the requirements are:

$$performing(a_i, role^k_{i,j}) := ((Cap(a_i, role^k_{i,j}) > 0)$$
$$\wedge \ (RequireRes(a_i, role^k_{i,j}) \subseteq AvailableRes(a_i)))$$

For any agent $a_i \in A$, I have $Assign(role^k_{i,j}) = a_i$ if $perform(a_i, role^k_{i,j}) = 1$; otherwise, $Assign(role^k_{i,j}) = Null$. Moreover, each resource and task are exclusive to be shared, and for $\forall a_i, a_j \in A$ I require $AvailableRes(a_i) \cap$ $AvailableRes(a_j) = \phi$. Similarly, for tasks, $perform(a_i,$ $role^k_{i,j}) \wedge$ $perform(a_j, role^k_{i,j}) = false$.

Based on this joint intention coordination model, team coordination is defined as: $\Xi = INF \cup ROLE \cup RESOURCE$ and has been segmented into pieces $tc \in \Xi$, which is either a domain event, a joint activity or an exclusive resource. By allocating those coordination to specific agents, for example, by fusing all the preconditions of domain events to a single agent and allocating roles to capable agents with required resources, a $plan_{i,j}$ will be activated and implemented. Therefore, a subgoal g_i will be achieved.

2.2.2 Utility Function

The objective of team coordination is to achieve the top level goal G by performing as many sub-goals g_i as possible. Suppose that in a period of time, domain events sensed by A are written as $\Upsilon \subseteq INF$. Only part of $PLAN$ will be activated by Υ, which is written as $PlanValid \subseteq PLAN$. Therefore, $\forall plan_{i,j} \in PlanValid, plan_{i,j}.p_{i,j} \subset \Upsilon$. Then $Complete(plan_{i,j}) = 1$ if $\forall r \in plan_{i,j}.q_{i,j}, assign(r) \neq NULL$. This equation requires that all its joint activities be assigned to one of the team members. The objective function is to get reward from implementing all the roles $r \in plan_{i,j}.q_{i,j}$, where $plan_{i,j} \in PlanValid$.

Before I discuss how the reward for each role is defined, I will introduce two important parameters:

- $UsefulInf(r) \subseteq INF$ defines all the useful domain events which are not required but are helpful for performing r, e.g., knowing which street has been blocked is helpful for performing the role of driving a fire trunk.

- $UsefulRes(r) \subseteq RESOURCE$ defines all the non-requested but helpful resources to perform r, for example, a fireproof suit is helpful for fire fighting.

I define the utility to implement a role r as [1]:

$$\begin{aligned}
&U(r) = \\
&(Cap(Assign(r), r) + 1) \times reward(g_i) \\
&\times (\frac{|UsefulInf(r) \cap AvailableInf(Assign(r))|}{|UsefulInf(r)|} + 1) \\
&\times (\frac{|UsefulRes(r) \cap AvailableRes(Assign(r))|}{|UsefulRes(r)|} + 1)
\end{aligned}$$

In this formula, $r \in plan_{i,j}.q_{i,j}$ and $AvailableInf(Assign(r)) \subseteq INF$ are all the domain events previously known by the agent who is performing r. The function shows that the utilities of performing a role r depends on the rewards of achieving its sub-goal g_i ($reward(g_i)$). Moreover Higher reward is foreseen if r is assigned to an agent who is more capable of performing that role than others, holds more useful resources, or knows more helpful domain events. The team coordination objective function is defined as:

$$maximize(\sum_{r \in ValidRoles} U(r) \times d^{|t(r)|})$$

where $ValidRoles = \bigcup_{plan \in PlanValid} plan.q_{i,j}$, d is an discount factor and $d^{|t(r)|}$ is the time discount factor before r is allocated.

[1]In my coordination algorithm, the cost of performing a task is ignored and the reward of performing a task is equal to that of allocating a task.

2.3 COORDINATING WASMS: AN EXAMPLE

My thesis is a part of the project of Coordinated Wide Area Search Munitions which is to coordinate large groups of Wide Area Search Munitions (WASMs) [81]. WASMs are a cross between an unmanned aerial vehicle and a standard munition. The WASM has fuel for about 30 minutes of flight, after being launched from an aircraft. The WASM cannot land, hence it will either end up hitting a target or self destructing. The sensors on the WASM are focused on the ground and include video with automatic target recognition, ladar and GPS. It is not currently envisioned that WASMs will have an ability to sense other objects in the air. WASMs will have reliable high bandwidth communication with other WASMs and with manned aircraft in the environment. These communication channels will be required to transmit data, including video streams, to human controllers, as well as for the WASM coordination.

The concept of operations for WASMs are still under development, however, a wide range of potential missions are emerging as interesting[20, 25]. A driving example for our work is for teams of WASMs to be launched from AC-130 aircraft supporting special operations forces on the ground. The AC-130 is a large, lumbering aircraft, vulnerable to attack from the ground. While it has an impressive array of sensors, those sensors are focused directly on the small area of ground where the special operations forces are operating making it vulnerable to attack.

The WASMs will be launched as the AC-130s enter the battle space. They will protect the flight path of the manned aircraft into the area of operations of the special forces, destroying ground based threats as required. Once an AC-130 enters a circling pattern around the special forces operation, the WASMs will set up a perimeter defense, destroying targets of opportunity both to protect the AC-130 and to support the soldiers on the ground. Even under ideal conditions there will be only one human operator on board each AC-130 responsible for monitoring and controlling the WASMs. Hence, high levels of autonomous operation and coordination are required of the WASMs themselves. However, because the complexity of the battlefield environment and the severe consequences of incorrect decisions, it is expected that human experience and reasoning will be extremely helpful in assisting the

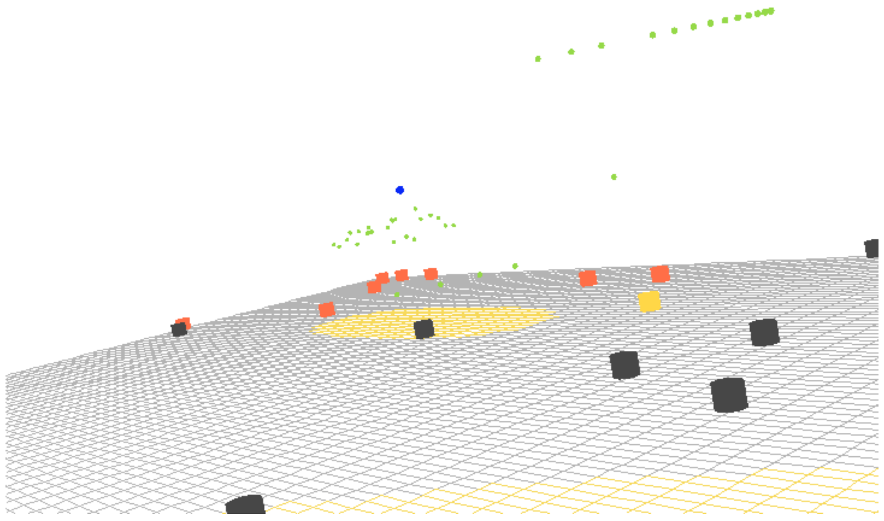


Figure 2: A screenshot of the WASM coordination simulation environment. A large group of WASMS (small spheres) are flying in protection of a single aircraft (large sphere). Various SAM sites (cylinders) are scattered around the environment. Terrain type is indicated by the color of the ground.

team to archive its goals safely and effectively

Many other operations are possible for WASMs, if issues related to coordinating large groups can be adequately resolved. Given their relatively low cost compared to Surface-to-Air Missiles (SAMs), WASMs can be used simply as decoys, finding SAMs and drawing fire. WASMs can also be used as communication relays for forward operations, forming an ad hoc network to provide robust, high bandwidth communications for ground forces in a battle zone. Since a WASM is "expendable", it can be used for reconnaissance in dangerous areas, providing real-time video for forward operating forces.

While my domain of interest is teams of WASMs, the issues that need to be addressed have close analogies in a variety of other domains. For example, coordinating resources for disaster response involves many of the same issues [46], as do intelligent manufacturing [72] and business processes. The problem of coordination I am dealing with here can be informally described as determining who does what at which time, and with which shared resources and information.

3.0 STATE OF THE ART

In this section, we will survey previous research to the multiagent coordination problem. Although they cannot be applied directly to solve large-scale multiagent team coordination when considering some improper assumptions or limitations, some of their ideas are critical to my research. There are three major groups of existing research: approaches according to BDI models, centralized coordination models and decentralized coordination models.

3.1 TEAMWORK

3.1.1 Belief-Desire-Intention model

The Belief-Desire-Intention (BDI) Model [73, 13] is the most popular architecture for practical reasoning agents. It combines basic ideas from human psychology with formal logic and architecture in the domain of intelligent agent applications. Intelligent agents following the BDI model are made up of the following three parts: Belief defines an agent's local knowledge base that represents what the agent "knows"; Desire defines the goal that the agent is trying to achieve; and Intentions define the agent's currently adopted plans of how the goal will be achieved. Plans are predetermined sequences of actions (or sub-goals) that can be specified as accomplished. Tasks are combinations of actions that achieve certain outcomes or responses to events and are used by the agent to further its intentions.

In the BDI model, agents act in both reactive and deliberative ways. When an agent detects a system event, it looks for relevant plans that respond to this type of event; then, for each relevant plan, the agent examines the appropriateness of the plan to the situation in which the agent finds itself; after that, the agent selects and starts executing the most appropriate plan. Agents also do deliberative planning: what goal to pursue, or alternatively, what event to react to; how to pursue the desired goal; and when to suspend or abandon the goal and change to another goal.

The agent may also vary its balance between reactive and deliberative behavior by changing the amount of time allowed for deciding what to do next. This enables the agent to be more or less sensitive to changes in the environment, that is, to be more or less "committed" to its current plan.

BDI is similar to the human decision process, and it is easy to be understood and implemented. But it also inherits the drawback of human decision where its model is hard to be quantitatively measured. There are two reasons [67]. The first reason is that the BDI model is incapable of handling uncertainty and communication cost in the environment of real domain. For example, Joint Intention theory ignores the communication cost to obtain the mutual belief [67]. Second, the BDI model ignores the computation complexity for a decision. Therefore, the BDI model cannot direct individual agents to act toward their joint goal while maximizing team performance, and the efficiency and complexity of teamwork cannot be guaranteed. Furthermore, most BDI models, such as STEAM, depend on a precise and detailed model to make decisions, which is not feasible for scalable coordination.

When the BDI model extends to multiagent teamwork, there are two branches of research: the Joint Intention model and the Shared Plans model.

3.1.2 Joint Intention Model

In the Joint Intention model, all team members jointly intend a team action if they jointly commit to complete it and mutually believe that all of them are performing it [89]. There are three main points in this model [92]:

1. All members mutually believe that a goal is not achieved.
2. All members have committed it as their mutual goal.
3. All members believe that it will be eventually achieved and will hold until it is achieved.

GRATE is the first implementation according to joint intention model [43]. This system utilizes the joint intention model to establish a collaborative protocol and monitor the execution of joint activity. The key to the success of this approach, is the explicit, detailed model that each agent has of the joint activity and of other members of the team. Agents use these models to reason about actions that will aid the achievement of joint goals.

3.1.3 Shared Plan Model

Unlike Joint Intention model, the Shared Plan model is based on intention attitude, other than joint mental attitude [89]. Grosz [37] defines a team member's intention as "directed towards its collaborator's action or towards a group's joint action and is a set of axioms that guide an agent to take actions, including communicative actions that enable or facilitate its teammates, subteam or team to perform assigned tasks" [38]. Shared Plans model a goal that will be realized in a set of social plans. Each of the social plans can be decomposed into joint activities which are specifically realized by one or several single agents with constraints of temporal or causal ordering.

Therefore, the Shared Plan model decomposes team activities hierarchically into multiple levels of actions. Those actions are comprehensive in handling of the partiality of belief and intention for only single agents or partial individual team members. We call it as sub-plans [85]. For example, in robot soccer, "winning the game" is a high-level plan for the intention and belief of all robots. This plan can be realized with two sub-plans: attacking and defending. Not all team members are required to commit the intention for a sub-plan, for example, goalkeeper is not required to attack.

3.1.4 STEAM

Tambe combined the Joint Intention model and the Shared Plan model and developed a general model of teamwork for persistent coordination in an uncertain, complex, and dynamic environment [89]. Unfortunately, its coordination requires a precise model of each individual agent.

In this model, joint intention is built within the block of teamwork, and shared plan model is used to build hierarchical structures between blocks, such as joint intentions, individual intentions and beliefs of the teammates. Therefore, in STEAM, the main purpose of shared plan model is to carry the domain knowledge and the framework of how agents would act and cooperate with the others. Specifically, there are two separate hierarchies [93]: Team Organization Hierarchy, which is building subteams and defining role assignments; and Team Activity Hierarchy, that refines team operations and team plan reactions such as pre- and post-conditions of a plan. Under the hierarchical structure set up by the shared plan model, joint intention theory within blocks drives communication to obtain mutual beliefs and mutual commitments for setting up or voiding of joint intentions. The purpose is to build a reasoning model of communication that is also used for monitoring team performance and detecting conflicts. This model is domain independent.

3.1.5 Team-Oriented Programming

Tidhar [95] used the term "team-oriented programming" to describe a conceptual framework for specifying team behaviors based on mutual beliefs and joint plans, coupled with organizational structures. His framework also addressed the issue of team selection [95]. Team selection matches the "skills" required for executing a team plan against agents that have those skills. Jennings's GRATE* [42] uses a teamwork module, implementing a model of cooperation based on the joint intentions framework. Each agent has its own *cooperation level* module that negotiates involvement in a joint task and maintains information about its own and other agents' involvement in joint goals. The Electric Elves project was the first human-agent collaboration architecture to include both agents and humans in a complex environment [18]. COLLAGEN uses a proxy architecture for collaboration between a single agent and user [76]. While these teams have been successful, they have consisted of at most 20 team members and will not easily scale to larger teams.

3.2 CENTRALIZED COORDINATION ALGORITHMS

Centralized coordination algorithms by definition require that one agent is a special actor. It is responsible for the whole decision process and produces the joint policy while the rest of agents only act as executors [32]. Typical assumptions for centralized control algorithms are: complete communication with central agent, e.g., message board [15], or central information agents have complete knowledge of the team [23]. Unlike decentralized approaches, which seek "satisfied" strategy in trade of uncertainty, centralized coordination approaches focus on searching for an optimal strategy.

While there is not yet definitive, empirical evidence of the strengths and weaknesses of each type of architecture, it is generally believed that centralized coordination can lead to behavior that is closer to optimal, but more distributed coordination is more robust with failures of communications and individual nodes [14]. The major issue when applying a centralized coordination approach to scalable coordination is that the communication limitations and computation complexity when a team scales up are ignored, which makes it unfeasible for our domain.

To find the optimal joint actions in centralized coordination models, there are several approaches listed below.

3.2.1 Classical Coordination

The traditional model of centralized coordination starts from small team coordination and is defined as a planning problem. There are three major assumptions: Agents are acting independently; they have complete knowledge of the world; and their activities are deterministic. Therefore, agents have the total knowledge of how an action will affect a state to be transited to another state without the inferences of the other team members. The solution of a classical coordination problem can be encoded in a propositional language, e.g., STRIPS [31]. By defining the relationship between world state, preconditions and execution outcome, classic coordination algorithms typically perform an operation to search in the action space of the domain. The objective is to find the sequence of activity for each agent who will jointly transition the original team state to the target state with the least cost. Therefore, by designing search algorithms in a given data structure, such as a plan graph [10], the size of the team has been extremely limited when the algorithm is applied in complex coordination.

Classical centralized coordination requires a precise model of the world, where uncertainty in the system has to be ignored. The state space is also extremely limited so that the search algorithm is computationally feasible to be performed. Moreover, designing as a planning problem loses the flexibility of coordination, where agents are not capable of reacting quickly to frequent dynamic environmental changes.

3.2.2 Market-Based Coordination

The market-based approach is very popular in multiagent society for coordination problem solving [5]. In market-base applications such as TraderBots [26] and Tracer [30], both tasks and resources are treated as merchandise. One agent acts as auctioneer, and other agents act as bidders. Agents bid for either single items or combinatorial sets of items in order to maximize their own utilities. The auctioneer maximizes its own utility by "selling" its "merchandise". Winner determination algorithms [78] are used to determine the allocation of tasks and resources by the auctioneer. Because of the central position of the auctioneer, via seeing the bids, it develops a complete knowledge of how agents will use a task or resource if allocated to them. Thus, the auctioneer is able to perform assignments that maximize team utility.

This research turned out to be very efficient for complex task and resource allocations, but it is not scalable [103]. Moreover, the market-based coordination approaches are not fit for information sharing or for sensor fusing. Other market-based approaches include negotiation [28] via contract nets [106], or local optimal approaches, such as cluster or statistic methods [27].

3.2.3 Hierarchical Coordination

Another group of approaches coordinate agents with a hierarchical organization [109, 24]. In this multiple-layered architecture, the high-level module is responsible for domain problem solving and high-level goal decomposition, while the low-level module is responsible for the detailed analysis and temple sequencing coordination, which are more likely to be domain-independent and are easy for code reusing [109]. For example, Beard proposed a three-levels hierarchical control strategy for a constellation of spacecraft [7]. In the highest level, agents respond to the high-level issues of mission directives to all spacecraft. The second level responded to decisions based on agents' limited information, which was the position of each spacecraft. The lowest level which is domain-independent implemented simple control strategies to move each individual spacecraft to avoid collisions.

There are two major bottlenecks when hierarchical coordination strategies are applied to coordinate a large team. First, hierarchical coordination integrates some kind of centralized control, and it may not capable in some domains that are featured as ad-hoc. Second, when the team size increases, more layers are required to be added in the hierarchy, but there is a trade-off between adding layers and the computational complexity [7].

3.2.4 GPGP/TAEMS

Generalized Partial Global Planning (GPGP) and its associated hierarchical task network representation, TAEMS (Framework for Task Analysis, Environment Modelling, and Simulation) was first put forward by Lesser [47]. GPGP approach used the TAEMS tasks structure with two major innovations: partial global planning (PGP) framework [22] for domain-independent coordination making; and a dynamic evolving goal tree for agents' distributing search for their own coordination policy. The advantage of TAEMS is its capability in handling uncertainty in the environment with the module called Design-to-Criteria (DCT) Scheduling [97]. Unlike the GPGP model, which is more suitable for scheduling, TAEMS can perform dynamic planning or coordination. In DCT, online scheduling uses a heuristic algorithm to build a plan or schedule to meet agents' dynamic goal criteria. The key to this algorithm is to generate optimal MDP policy by heuristically searching the hierarchical task network or dynamic hierarchical goal tree.

In contrast to STEAM, which is based on the BDI model, GPGP is focused on optimization to maximize overall team utility. Unfortunately, GPGP/TAEMS approach requires a centralized reasoning framework, which is computationally difficult to using in coordinating a moderate or scalable team [47].

3.3 DECENTRALIZED COORDINATION ALGORITHMS

In a decentralized coordination model, agents are highly distributed and share knowledge, but must independently make their individual decisions. This is the basic model for scalable coordination.

3.3.1 Multiagent Markov Decision Process (MMDP)

In the decision theoretic model, agents in a multiagent team individually or globally generate their policies based on their own observation and knowledge of the team. The objective is to yield the best activities to maximize the team performance.

When the MDP model is applied to multiagent coordination, each agent has its own set of actions, a given common goal has to be solved by cooperation, and all agents have a common utility function. Boutilier [11] models this problem as a multiagent Markov decision process (MMDP), which is a tuple: $< S, \alpha, \{A_i\}_{i \in \alpha}, T, R >$. Unlike the definition of a single agent decision process, S is defined as a finite set of team states which merges all states of single agents in the team. α is the set of agents. $A = \times_{i \in \alpha} A_i$ is the set of joint actions. At any stage of the process each agent will select an individual action to perform. A_i is a finite set of actions available to agent i. $T : S \times A_1 \times A_2 \times ...A_n \times S \rightarrow [0, 1]$ is the transition function where a team state will be transferred by agents' joint activities. If only one agent is responsible for finding the policy of joint action A for all the agents, $S \rightarrow A$, this agent is acting as controller and all the other agents are only executors. It is a centralized approach [11].

In the decentralized decision process, each agent in the team knows the state of the team exactly, and it will execute the action according to its own policy, $\pi_i : S \rightarrow A_i$. The objective of decentralized coordination is to search local policies π_i to maximize the joint reward. As the nature of MDP, agents can fully observe the team state. But an agent's single activity depends on how the other agents act. To solve this, MMDP enforced a strict social convention that is known to all agents in the initial stage. Following this research, Koller put forward a factored MDP model where the global reward function is represented by a Dynamic Bayesian Networks and optimal policy can be searched through learning programming along with the team's social order [39]. The major difficulty of applying MMDP in scalable teamwork is that no agents has the complete knowledge of the team state.

3.3.2 Decentralized Partially Observable MDP (DEC-POMDP)

The general decentralized coordination model for scalable teamwork assumes that agents are widely distributed in an open environment with limited communication. In addition, because of the nature of the environment, agents are not able to access the complete team states and only have a partial view of the external environment. Therefore, MMDP's assumption of complete knowledge and observation for all agents is not met.

DEC-MDP is generalized from the MMDP model if the agents in the team only perceive a partial state of the environment [101]. DEC-POMDP is modelled by Bernstein as a tuple $< S, \alpha, \{A_i\}_{i\in\alpha}, O, \Omega, B, R >$ [74]. In addition to MMDP, Ω is the finite set of observations from single agents. $O : S \times \Omega \rightarrow [0, 1]$ defines the observation function to map the observation probabilities that an observation could be made in a given state. $b_i^t \in B_i$ defines agent i's observation history until t where $b_i^t =< o_i^1, ..., o_i^t >$. Policy for agent i, $\pi_i : B_i \rightarrow A_i$ maps agent's current beliefs to an action which the agent believes to benefit the team most.

DEC-POMDP requires a tight social convention, because how one agent computes its expected value function depends on the choices of other agents. For example, in [39], an exhaustive algorithm for finding the joint utility function of n agents is to recursively find the optimal joint utility function for $n - 1$ agents and, for the other agent, to find the individual activity which maximizes the joint team reward of those n agents. The last agent to generate its expected utility function knows enough to select its optimal action. This action choice is propagated through the rest of the agents in reverse order, allowing each agent to select its optimal action. Unfortunately, this does not hold for scalable coordination when agents cannot fully observe the activities of all the other agents.

Although the DEC-POMDP model is more realistic for handling uncertainty in scalable teamwork, Bernstein provided the mathematical evidence that decentralized decisions to find the optimal joint policy are NEXT-complete [9]. To find practical solutions, DEC-POMDP researchers either focused on localized optimizations [62], or restricted the domain where extra assumptions can be made, such as transition independence or collective observability [8]. All those solutions cannot be convinced to be a general solution for scalable coordination.

3.3.3 Distributed Constraint Satisfaction

Distributed Constraint Satisfaction Problems (DisCSPs) are another field of research that has been frequently applied to task assignment, resource allocation, and mobile sensor network [21, 71]. The multiagent team in DisCSP is defined as a constraint network, and each agent holds its own local constraints. Agents are connected by constraints among variables of different agents. Yokoo [107] modelled DisCSPs as variables $\{x_1, x_2, ..., x_n\}$ from discrete domains $\{D_1, D_2, ..., D_n\}$. Each constraint k is defined according to a predicate $p_k(x_{k1}, x_{k2}, ..., x_{kj})$, which is a Cartesian product of $D_{k1} \times ... \times D_{kj}$. The predicate is true when assigned value $x_{k1}, x_{k2}, ..., x_{kj}$ satisfies the constraints. To solve a DisCSP, agents have to find an assignment of values for all variables to satisfy all the constraints. In practice, agents assign values to their local variables and generate a locally consistent assignment. By exchanging messages about their setting to local constraints, agents check the value assignments for local consistency with one another. By changing the value assignment when inconsistencies are detected, this process continues until all global constraints are satisfied [108].

The most trivial solution is to select one agent to have all the variables, domains and constraints. This centralized DisCSP problem has been converted into a centralized one which is not applicable to many domains [107]. Decentralized solutions of DisCSPs have been split into two branches [56]: Synchronous Backtarcking (SBA) and Synchronous Forward-Checking (AFC). SBA maintains a single backtrack process at any time. Agents are assumed to maintain a complete social order for variable assignments. Agents get information about assignments to other agents that are ahead of them in the total order. When receiving a message, agents try to solve their local assignments in an manner that is consistent with the assignments received and a 'No Good' message will be sent when any conflict cannot be solved. In this case, the previous assignments have to be changed, and a new process will start. On the other hand, AFC maintains multiple concurrent processes with the data structure of Current Partial Assignement (CPA), which is more suitable for scalable coordination. When each agent gets a CPA, it will perform a local assignment and forward the update CPA to another agent if the consistency can be solved. Otherwise, it will send the original CPA back to the agent that it came from.

Several researches on enhancing the solutions of DisCSPs have been done. For example, a recent research called Concurrent Backtracking combines both methods and runs multiple backtrack search processes asynchronously [110]. But DisCSPs model still does not fit the nature of our domain. The major weakness is that DisCSPs are low efficient to be solved in trading of finding optimality. When requiring social orders, agents in DisCSPs will be very slow and most of their applications cannot meet real-time control. All constraints are supposed to be kept unchanged from the initial stage, but in the emergent military domains, constraints are dynamic and change frequently. For example, if a WASM is killed, all the other WASMs will have to change their local constraints. Moreover, under uncertainty, agents cannot fully explore their local variables. For example, a WASM does not know it cannot follow a planned path to fly until it discovers a hostile SAM missile is deployed on its way points.

3.3.4 Network Distributed POMDPs

Although DEC-POMDP and DisCSP have been two major approaches to solve distributed coordination, they have different advantages and disadvantages. DEC-POMDP is good at real-time control for handling uncertainty, but it is unable to exploit the locality of interaction [70]. DisCSP is good at exploiting the locality of interaction, but as explained above, it is incapable of uncertainty. Nair made an initial effort to combine the two approaches in a way that can take the advantages of both [70]. The constraints around the teams are explicitly represented as a network structure. The algorithm called LID-JESP uses dynamic programming of POMDP based on offline distributed policy generation, and can perform optimal policy search on the tree-structured agent-interaction graphics. Unfortunately, no explicit theory or experiment results are shown that this algorithm is capable of coordinating a team with more than 10 agents.

3.3.5 Swarm Intelligence

Swarm-based approach is a typical heuristic coordination algorithm [77, 2, 16]. It provides a distributed, highly scalable way to perform coordination. Swarm agents design is inspired by the examples of collective behavior exhibited by biological systems, such as social insects, and the swarming, flocking, herding, and shoaling phenomena in vertebrates. Swarm multi-agent system is self-organizing and can construct collective (or macroscopic) behavior by emerging from individual (or microscopic) decision making. Swarm algorithms rely directly on locally sensed stimuli to adjust thresholds and make decisions [2], while in a large team, agents may use arbitrary information obtained locally or from other agents. This additional level of flexibility can be leveraged for better performance through synergistic interactions with the other algorithms that are presented.

3.4 MULTIAGENT COMMUNICATION DECISION PROBLEM

MMDP has the view of a complete team state, while DEC-POMDP assumed that agents' belief will be updated merely from partial observation. Pynadath [67] and Xuan [105] argued that when a team undertakes a complex task, agents can make better decisions by sharing their individual beliefs through communication. Therefore, by sending messages to teammates with a belief that they did not observed, the team can obtain a higher reward. When assuming that communication is not free, agents have to make decisions on how to target their communication to share those individual beliefs. This problem is modelled as Multiagent Communication Decision Problem. Xuan [105] extended the MMDP so that communication is an explicit action in A_i and will improve the joint reward, but a cost will be occurred when a message is sent. By integrating communication decisions, MMDP can be more efficient in handling environmental uncertainty.

Goldman [35] developed a practical approximation approach to optimize information exchange for small teams. In her approach, similarly to the research above, agents make decisions considering both standard action and communication as the choices of optimal activity to maximize the global utility function. Her key algorithm is based on myopic meta-level control of communication [36] that balances the trade-off between the cost of communication and the value of information to increase the global utility function. Although this solution allows for online decision making, it also requires an off-line optimal solution calculation beforehand. This off-line calculation is hard or impossible to carried out when Goldman's approach is applied to scalable coordination.

To reduce the communication decision costs in MMDP, Shen [86] proposed a technique to abstract the raw data by using a distributed Bayesian network. This technique incorporates a hierarchical action selection approach to define how and when the transition between abstraction data and raw data will take place in the communication process. This research is similar to the research presented here in making use of domain knowledge to further abstract communication for efficient communication decisions. Unfortunately, this approach requires a decentralized MDP algorithm to be constructed from a Bayesian network. The purpose is to find a joint communication policy to minimizes the expected communication cost, but it cannot be applied to large team coordination when agents' observation of the team state is incomplete.

Pynadath [67] generalized the DEC-POMDP model by extending a part of communication action. He called this model COMmunicative Multiagent Team Decision Problem (COM-MTDP), inspired by economic team theory [40]. The COM-MTDP model is represented as a tuple $< S, \alpha, \{A_i\}_{i\in\alpha}, \Sigma, O, \Omega, B, R' >$. It differs from DEC-POMDP by having two additional parts: communication action Σ , which will update individual agent belief; communication action, by updating agents' mutual beliefs about important aspects of their executions, which will improve agents' performances. Therefore by performing Σ, multiagent team can get a different reward R' other than R in DEC-POMDP where communication may get extra reward or incur cost.

In general, all the decision theoretic models require that agents make a decision based on the common utility function, which is either explicitly defined or learned from various algorithms, such as reinforcement learning. This technique requires that the team goal can be quantitatively measured. However, scalable-team coordination always involves a complex and high-level task with uncertainties in the environment, in which a goal cannot be precisely measured or defined. Therefore, optimal coordination for teamwork in real domains is impractical. For example, Pynadath agrees that his COM-MTDP model cannot be used for optimal coordination, although it is helpful for comparing different practical models and identifying feasible improvement [67].

3.5 TOKEN-BASED COORDINATION

The token-based approach was first introduced from networking design [1]. The basic idea is to encapsulate one or more types of critical controls into tokens. Tokens can be passed in parallel, and agents can only access an exclusive control when holding a token in which the control is embedded. For example, in network design [94], this exclusive control is the access to the network bandwidth.

3.5.1 Key-Based Coordination

The idea of token-based approach was introduced to multiagent coordination research by Wagner [88]. He renamed tokens as "keys" and applied them to the coordination of dynamic readiness and repair service in aircraft simulation. In his research, he used the TAEMS [47] as the team coordination. A centralized scheduler is required to subset the service team without any agent serving in one more different function subset. Each piece of coordination is encapsulated in a key and passed from agent to agent. Agents are organized in a circle. When an agent is holding a key for its coordination subset, it can declare its intended action or schedule; evaluate existing proposals by either confirming or denying them; put forward its own proposal; and read whether its own proposal has been rejected or accepted. The coordination of the key algorithm is heuristic, and the author cannot evaluate the optimality of the coordination decision process. Moreover, the author makes an assumption that complete communication is available for all agents, and the communication cost is ignored. But the most important aspect is that the existence of the centralized scheduler makes this approach difficult to apply to scalable coordination.

3.5.2 Token-Based Approach for Single Coordination Task

Recent research focusing on scalable coordination [64] illustrate that exponential search spaces, excessive communication demands, localized views, and incomplete information of agents pose major problems for large-scale systems. Scerri's initial work on token-based approaches promises a way to address these challenges [64], and the effectiveness of large-scale, token-based coordination has also been demonstrated in the Machinetta proxy architecture [83]. There are two major algorithms for single coordination tasks. LA-DCOP encapsulates independent tasks into tokens, and tokens are passed around heterogeneous teams with local constraints [80]. The key idea of this algorithm is that an agent can accept any task token that meets its local constraints but may reject or kick out current tokens if any other incoming token can increase its utility function. In [102], the information sharing algorithm encapsulates domain information in tokens. By exploring the relevance between pieces of information in domain knowledge, tokens are more likely to be routed to those teammates who can make use of the encapsulated information. Both of these algorithms have turned out to be scalable. However, they did not work together to enhance the team performance.

4.0 TOKEN-BASED COORDINATION FRAMEWORK

The objective of my research is to put forward a general model of scalable-team coordination with the major aspects of control distributed over a scalable heterogeneous team. In my thesis, I focus on three major aspects of coordination: role assignment, resource allocation, and sharing information. My algorithm is in order to coordinate a few hundred agents. The success of scalable-team coordination depends on how agents make individual decisions based on their incomplete knowledge of the team status with limited communication capabilities. My proposed research will model the general coordination decision problem as a token-based communication decision problem. The key to this model is how agents target their communications to send tokens in a way that maximizes team utilities. In this chapter, the basic framework of token-based coordination is introduced first. There are four parts to this framework. First, every piece of team control is encapsulated into tokens. Second, the data structure of tokens according to my scalable teamwork model will be described. Third, I will describe the token-based team organization that limits each agent's direct communication to just a few of its teammates, which is similar to the organization of human society. Agents can be organized as different social network topologies, which vary in efficiency. Moreover, corresponding to different activated plans, the terminology of sub-teams will be defined where agents can be organized as different sub-teams based on the associate network. Fourth, my initial design of a local decision model built from previously incoming tokens is presented.

4.1 ENCAPSULATING TOKENS

Based on recent successful token-based algorithms for task allocation and information sharing, my first idea is that *all* coordination interactions, including sharable information, assignable tasks, and sharable resources are encapsulated into *tokens*. The agent holding the token has exclusive control over whatever that token represents, such that tokens provide a type of access control. Agents may either keep a token or pass it on to teammates. For example, an agent holding a resource token has exclusive access to the resource represented by that token and passes the token on to another agent to transfer access to that resource. The resulting movement of tokens implements the coordination by distributing information, resources, and tasks to maximize the team reward with low communication overhead.

4.2 REPRESENTING TOKENS

Following the success of teamwork research, I have designed my scalable teamwork across the joint intention model and shared plan model. A $Token$ is defined as a data structure of communication *message* with required parameters including time stamp, threshold, path, and initiation ID. It encapsulates everything that can be shared in team A. The structure of any token Δ_j is written as $\Delta_j =< ID, tc, path, threshold, TimeStamp >$, where $tc \in \Xi$ is a piece of coordination control, which is a role, an exclusive resource, or a piece of information according to the definition in Section 2.2.

Due to the nature of tc, Δ_j cannot be duplicated or re-sent. When an agent is holding Δ_j, it takes over control $\Delta_j.tc$. The agent will release $\Delta_j.tc$ if Δ_j is passed. To mark the uniqueness of coordination in the team, I require $\forall \Delta_i, \Delta_j, if\, \Delta_i \neq \Delta_j, then\, \Delta_i.tc \neq \Delta_j.tc$. Specifically, if $\Delta_j.tc \in INFO$, I call Δ_j an information token and, to be clear in my presentation, write it as Δ_j^I; I call it a role token if $\Delta_j.tc \in ROLE$ and write it as Δ_j^R; and I call it a resource token if $\Delta_j.tc \in RESOURCE$ and write it as Δ_j^S.

$\Delta.path$ records the sequence of agents where Δ_j has been passed. $\Delta.path$ is also used as a stop condition for information and role tokens if $|\Delta.path| > Length$. $Length$ is pre-defined as the number of agents that Δ is allowed to be passed to before it is stopped.

Threshold generalizes a threshold for resource and role tokens that can be accepted by any agent, but is not required for information tokens. An agent may keep a resource if its desire for that resource is greater than the token's *threshold*. Determining this requirement is beyond the scope of my thesis. While an agent holds a resource token Δ, $\Delta.threshold$ slowly increases. This mechanism ensures that resources can flow through the system rather than accumulate at a few points. When the resource token is being passed, $\Delta.threshold$ is decreased to avoid a situation in which a token would be passed indefinitely. For example, to coordinate a team of WASMs, all air spaces are represented as resources. A WASM holding an air space resource token will be allowed to access the region that the token represented, but it might be forced to relinquish the token and free the region as its threshold increases. This would return access to the region that the WASM has already traversed to the general community. Similarly, a role token Δ will be accepted by an agent whose capability is greater than $\Delta.threshold$. If Δ is not accepted by the current agent, $\Delta.threshold$ will be slightly decreased so that it can be easily accepted by the other agents. This mechanism will avoid a situation in which a role token might linger infinitely within the network in order to find a highly capable agent to perform it.

The basic algorithm for token routing is Algorithm 1: At each time point, agent a_i will wait for incoming tokens, which are defined as $Tokens(a_i)$ (line 2). For each incoming token Δ_j, if the token does not reach its stop condition (line 4), agent a_i will decide whether this token can be accepted (line 5). If it is, a_i will keep it (line 6) and will to pass it to one of its associates called Next (line 9). Otherwise, before passing it, agent a_i will add itself in the token's path (line 8).

Algorithm 1: Decision process for agent a_i to pass incoming tokens.

```
1: while true do
2:   $Tokens(a_i) \leftarrow getToken()$;
3:   for all $\Delta_j \in Tokens(a_i)$ do
4:     if $|\Delta_j.path| < Length$ then
```

```
5:         if $Acceptable(a_i, \Delta_j)$ then
6:           $Keep(\Delta_j)$
7:         else
8:           $Append(self,\ \Delta_j.path)$;
9:           $Next \leftarrow ChooseNext()$;
10:          $PassToken(Next, \Delta_j)$;
11:        end if
12:      end if
13:    end for
14:  end while
```

4.3 TEAM ORGANIZATION FOR SCALABLE COORDINATION

To make correct decisions, agents are required to obtain sufficient knowledge of the team state to support their decisions. This in turn requires the multi-agent team to create overwhelming messages, which should be avoided in the algorithm. It has been observed that, in a human group, members typically maintain a small number of acquaintances but can rapidly transmit information to any member of the group in a series of hops, a phenomenon known as a *small world effect* [100]. The most popular manifestation of this phenomenon is the *six degrees of separation* concept, articulated by the social psychologist Stanley Milgram [58]. Milgram concluded that there is a path of acquaintances with a typical length of six between any two people in the United States. This experiment showed that by using very vague (and often incorrect) information about other members of the population, people will pass a message along to someone who is better placed to find the intended recipient until the information reaches the desired recipient.

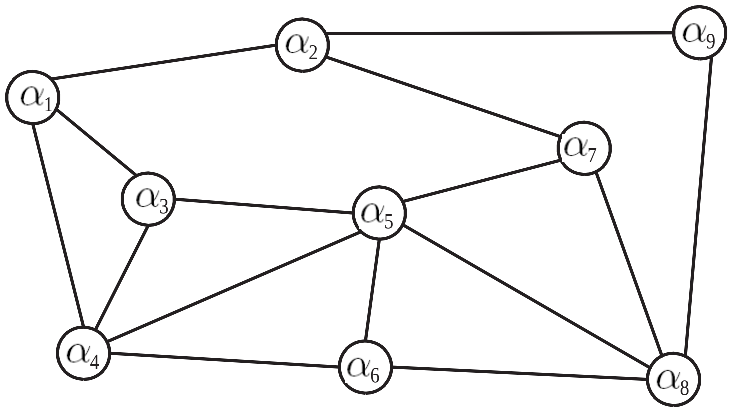


Figure 3: An Example of a sub-set of a typical associate network where each agent only has a small number of associates.

4.3.1 Associate Network

In my research, a logical network topology across the team limits agents to forwarding tokens to their neighbors in a network that I call an *associate network*. As a result, an agent receives tokens directly only from a small number of neighbors in the network and can thus build better models of those agents. Based on these characteristics, I can define my team organization model for large-scale teams. The *associate network* is an indirect graph $G = (A, N)$, where A is the team of agents and N is the set of links between any two agents. Specifically, for $a_i, a_j \in A$, $< a_i, a_j > \in N$ denotes that a_i and a_j are associates and are able to exchange tokens directly with each other. $n(a)$ is defined as all of the associates of agent a. Note that $n(a) << |A|$. A sub-set of a typical associate network for a large team is shown as Figure 3. In the figure, each node represents an agent member in the team and, when pairs of agents are connected by a line, they can directly exchange tokens with each other.

4.3.2 Social Network Topology

As noted by social scientists, communication between people is impacted by network topology. In my thesis, agents that pass tokens among a large-scale team adopt the same manners as a social group composed of human beings.

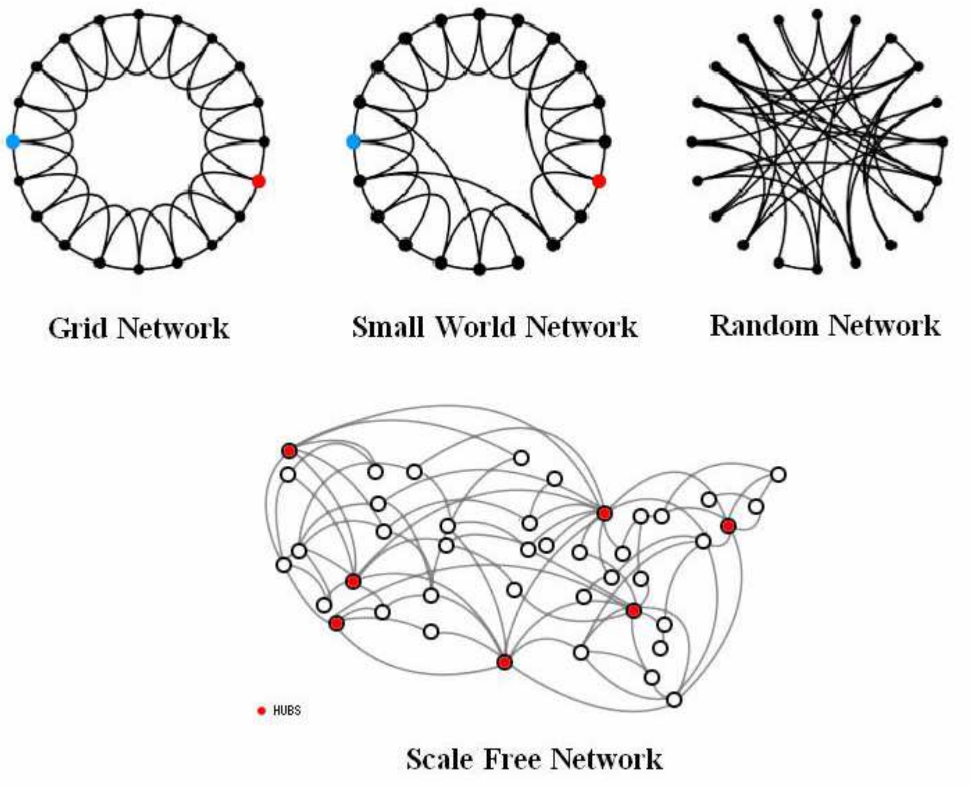


Figure 4: Examples of the four social network topologies: Grid, Small World, Random, and Scale-Free.

The properties of social network structure have been comprehensively studied [63]. According to this research, there are several parameters that are important in understanding or predicting the behavior of token-passing among large-scale teams. Key factors include the small world effect, degree distributions, clustering, network correlations, random graph models, models of network growth and preferential attachment, and dynamical processes taking place on networks [63]. Most of these factors are interrelated. In my thesis, I specifically focus on only three properties: average distance, degree distribution, and the average number of associates.

- Average distance: (commonly studied as "small world effect" [100]). The average distance $l = \frac{1}{\frac{1}{2}n(n+1)} \sum_{a_i, a_j \in A, i>j} distance(a_i, a_j)$, where $n = |A|$ and $distance(a_i, a_j)$ represents the minimum number of agents a_i, a_j that a token must pass through one agent to another via associate network. For example, if agent $a_1$and a_2 are not associates but share an associate, $distance(a_1, a_2) = 1$.
- Degree distribution (commonly studied as "scale free effect") is the frequency of agents possessing different numbers of associates. The distribution can be represented as a histogram where the bins represent a given number of associates and the size of the bin is how many agents have such a number of associates [3].
- Average associates are the average number of associates that agents have on a team. This value can be used to infer how many choices agents may have when delivering a message.

Well-known types of social networks can be described using these properties. For example, a *random network* has a "flat" degree distribution while a *grid network* is distinct in that all nodes have the same degree (e.g, four is the only degree in a two- dimension grid network). A *small world network* [3] and a *scale-free network* [6] are two important types of social network topologies, and research has shown that each possesses some interesting properties. Small world networks have a much shorter average distance than regular grid networks. We hypothesize that the low average distance will improve the efficiency of token-passing because information can potentially take fewer "hops" to reach a defined destination.

A scale-free network, shown in Figure 4, is a specific kind of network in which the degree distribution forms a power-law, i.e, some nodes are very connected hubs and connect to other

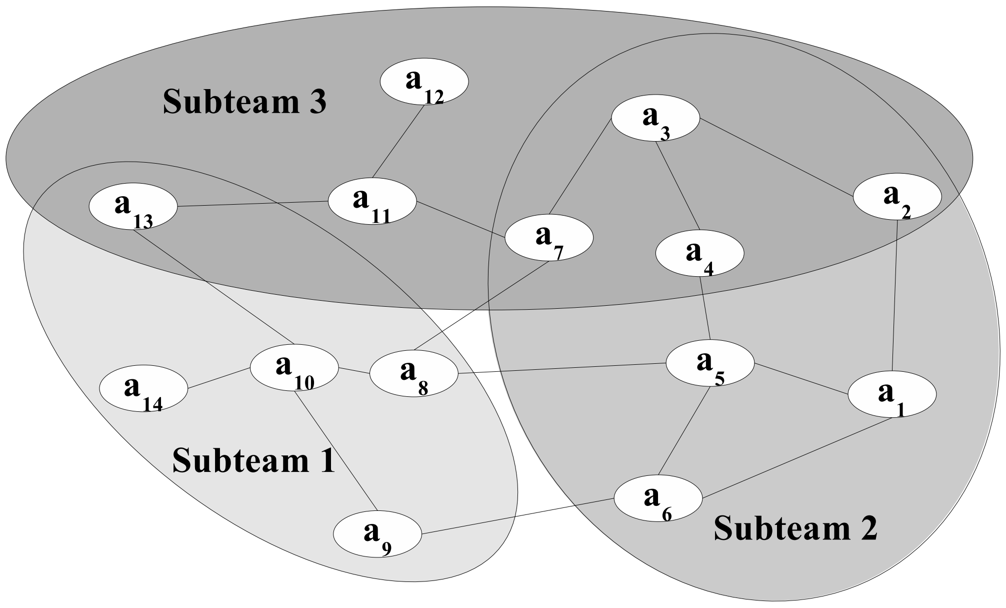


Figure 5: Relationship between sub-teams and the associate network

nodes much more than ordinary nodes. The hubs in scale-free networks give the advantages of centralized networks in which the distribution provides the advantages of centralized approaches. In the next chapters, I will investigate how different network topologies influence token movements.

4.3.3 Sub-Teams

Based on the concept of the associate network, we defined $Sub - team_i$, which includes any agents working on $plan_i$ *and* their neighbors in the associate network. The identities of the agents involved in role allocation are captured with $allocate(plan_i)$. In the case when either a conflict or synergy is detected, all but one of the plans must be terminated. The domain-specific knowledge of a termination of a plan can be defined as $^{term}recipe_i$.

Although individual agents commit the team to a sub-goal, it is a sub-team that will realize the sub-goal. The sub-teams formation process commences when an individual agent detects all of the appropriate pre-conditions that match a plan template in the library and subsequently instantiates a plan, $plan_i$. For each of the $roles_i$ in $plan_i$. A role token then is created to be allocated to the team. Once it has accepted a role r in $plan_i$, the agent becomes a member of the sub-team of $plan_i$ and makes a temporary commitment to perform the role represented by the token. Note that agents can accept multiple tokens and therefore can perform more than one role and belong to multiple sub-teams. Since allocation of team members to roles may change due to failures or changing circumstances, the members of a sub-team may also change. One example of this is when a member decides to drop a role for a more suitable task. This will lead to the best use of team resources because team members will execute roles that they are most capable of doing.

All sub-team members, both agents performing roles and their informed associates, must be kept informed of the state of the plan, e.g., they must be informed if the plan becomes irrelevant. This maximizes cohesion and minimizes wasted effort. Typically $|sub-team_i| < 20$, although it may vary with plan complexity. Notice that, typically, $sub-team_i \cap sub-team_j \neq \emptyset$, where $i \neq j$.

4.4 LOCALIZED DECISION MODEL

Under the token-based coordination decision framework, I have converted the complex coordination decision problem into a series of communication decision problems that focus on where to pass tokens. Tokens, when defined within this teamwork framework, have decoupled the joint activities of a large team where no single agent has sole responsibility over any shared decision.

In my thesis, agents use local decision theoretical models to determine when and where to pass tokens around the associate network. When an agent passes a token to an associate, that exchange is used to refine both of their local models of the team. This local model is used in a decision theoretical way to determine whether and where to forward any token that an agent currently holds in order to maximize the expected utility of the team. Informally, by using their local decision models, agents infer which team member will either use the information, resource, or task represented by the token, or is in the best position to know who will. Then, tokens will be passed there so that team performance will be benefited most.

By leveraging the idea that tokens are always interrelated, a received token can be used to decide where to send another token if there is a relationship between these two. For example, understanding the relationship between "I am hungry" and "pizza" is helpful for an agent to pass a "pizza" if it previously heard a piece of information about "I am hungry." Therefore, *all* available tokens are used to create models of the team, specifically by using the movement of one token to inform the movement of other tokens. Although previous tokens reinforce agents to create a more precise model of where to pass Δ, I cannot guarantee that it will gather all the related tokens. Unlike MDP, which precisely chooses the activity to maximize the expected reward, my approach to individual agents making decisions is to set up a probability model. The objective is to guess where to pass the currently held token based on previously received tokens. Therefore, in the process of building the probability model based on previous tokens, related tokens are more likely to be routed to the agents with better models. This process, in turn, will train token routing to be more localized. The design of how to build and update agents' local decision models to pass tokens to one of their associates is addressed in the remaining chapters.

5.0 TOKEN MOVEMENTS

In this section, I investigate the theory of how tokens will be passed from agent to agent in an associate network. The baseline of this token movement is a random walk in which an agent randomly forwards the token to one of its associates. The objective of investigating the random walk is to verify the feasibility of my token-based approach, which is based on peer-to-peer communication. The feasibility is derived from the fact that the token does not need to visit all agents to deliver a coordination element, whether it is information, a resource, or a role. On the other hand, if tokens are being passed along a very short path, the control that is encapsulated in a token is unable to reach an agent that can make use of it.

In the first section, a qualitative analysis of the boundary that a token should move past or its TTL (time to live) is presented. Specifically, the theory of random walks is applied to determine a lower bound on the TTL. It is important to note that this analysis is based on the research of [79], which is not my major contribution, but it is indispensable in showing the feasibility of my approach.

In the second section, based on theoretical analysis, an initial experiment of random token movement based on both unbiased and biased probability models is presented. The results of this experiment show that, although an agent's local decision model is imprecise, its bias can be increased a little in order to move tokens into the right direction This greatly enhances the efficiency of random token movements.

In the third section, we theoretically measure token movements in the Markov Chain. Based on mathematical analysis, the tokens' random walk is bounded. Moreover, this analysis of tokens' random movement is extended based on the different network topology of the associate network.

In the last section, a set of experiments on token movement based on agents' local probability models are presented to support my hypothesis.

5.1 RANDOM WALK: BOUNDING TOKEN MOVEMENT

When given a graph and a starting point, a random walk selects one of its neighbors at random and moves to this neighbor. Then we select a neighbor of this neighbor at random and move to it, and so on. The "random" sequence of points selected in this way is a random walk on the graph [51]. If agents on the team randomly pass tokens to their associates in a given associate network until the tokens are accepted by some specific agent, the movement of the tokens is a random walk as well. Thus, a random walk is the baseline of our token-based coordination algorithm. In this section, we investigate the basic token movement based on a random walk.

The associate network can be described as a connected graph $V = (V, E)$ with n nodes and m edges. Given an associate network G with n nodes and m edges, a token's *random walk* starts at some node v_i of G and, at each step, moves to one of the associates of the current node. For example, if the token is at node v_j, it randomly moves to an associate of v_j with a probability of $1/d(v_j)$, where $d(v_j)$ is the number of neighbors of v_j in G.

The probability that, at step h, the token is at node v_i can be denoted as $P_h(v_i)$. The theory of random walks [51], states that if a walk starts from any node in an undirected connected graph G, $P_h(v_i)$ converges to $\pi(v_i) = d(v_i)/2m$, where $\pi(v_i)$ represents the probability that a token will be at node v_i at any particular time.

There are two measures in random walk theory that are helpful for the analysis of random token movement: *hitting time* and *commute time*. The hitting time $H(v_i, v_j)$ is defined as the expected number of steps before agent v_j is reached by a token that starts from agent v_i. The sum of $\kappa(v_i, v_j) = H(v_i, v_j) + H(v_j, v_i)$ is called the commute time, which is the expected number of steps in a random token movement starting at agent v_i. After agent v_j is reached, the token returns to agent v_i again.

We are interested in two propositions from random walk theory [79]:

Proposition 1 The commute time of a token starting at agent v_i and reaching v_j before returning to v_i can be estimated as:

$$\kappa(v_i, v_j) \geq 2m/d(v_i)$$

Proposition 2 If we choose agent v_i uniformly from the stationary distribution over agents V and $v_j \neq v_i$, then we should have $H(v_i, v_j) \geq (2m/d(v_j)) - 1$.

Let τ be the first time that a token starting at v_i returns to v_i and that σ is the first time that it returns to v_i after reaching agent v_j. [79] determined $E(\tau) = 2m/d(v_i)$ from the probability $\pi(v_i) = d(v_i)/2m$. If we define $E(\sigma) = \kappa(v_i, v_j)$, we should have $\tau \leq \sigma$. Therefore, the proposition $\kappa(v_i, v_j) \geq 2m/d(v_i)$ holds.

The main idea of this theory is that, if a token can start a random walk from agent v_j, the random movement will choose an associate of v_j uniformly and then reach other agents before it returns to v_j. From Proposition 1, [79] found that, for a given undirected connected associate network G, if we choose v_i uniformly from the set of associates v_j, the expectation of $H(v_i, v_j)$ is exactly one step less than the commute time from v_j to v_i. We have known that the hitting time $H(v_j, v_i) = 1$ and that the commute time $\kappa(v_j, v_i) \geq 2m/d(v_i)$. Therefore,

$$H(v_i, v_j) = \kappa(v_j, v_i) - 1 \geq 2m/d(v_j) - 1.$$

The following proposition holds if we choose v_i uniformly from the stationary distribution over V and $v_j \neq v_i$ [79]. Note that for an agent v_j and a uniformly chosen agent v_i, the hitting time of $H(v_i, v_j)$ is minimal when v_j is an associate of v_i. According to Proposition 1, we have $H(v_i, v_j) \geq \kappa(v_j, v_i) - 1$ and $\kappa(v_i, v_j) \geq 2m/d(v_i)$. Hence, a token's random walk from agent v_i to agent v_j and its subsequent return to v_i is bounded.

5.2 TOKEN MOVEMENT UNDER UNCERTAINTY: PRE-EXPERIMENT

To test potential random token movement within a large-scale team, I ran an experiment in which 800,000 agents are organized in a three-dimensional lattice [102]. One agent is randomly chosen as the source of a token, and another is randomly chosen as the sink for that token. A probability value is attached to each link, giving the chance that passing the token down that link will get it through the smallest number of links to the sink. In the experiment results shown in Figure 8, I varied the probability of sending the token down links that actually lead to an agent requiring the token (as opposed to sending it down links that moves the token further away) and measured the number of messages (or "hops", one message per token movement from associate to associate) required to move the token from the source to the sink. For example, for the "59%" setting, messages are passed along links that get closer to the sink 59% of the time and to links further from the sink 41% of the time. Figure 8 shows that the agents only need to move closer to the target slightly more than 50% of the time to dramatically reduce the number of steps that the token takes to reach the sink. This result is encouraging because it shows that I do not need to construct accurate and complex models for information sharing and that only reasonable models are needed to improve an agent's guessing. Thus, even relatively inaccurate models of associates are potentially capable of leading to efficient, targeted token delivery. The key question left in my thesis research is how to create models that allow the agent to "guess" correctly at probabilities greater than chance.

5.3 MODEL TOKEN MOVEMENT AS A MARKOV CHAIN

A random walk is a finite Markov chain and it is time-reversible [51]. For a specific token movement, we can define different states based on a Markov Chain. As the graph 7 shows, state s_i defines the state that the token moves to an agent with the shortest distance of i from the sink agent. Moreover, probability $P_{i,j}$ defines the probability of the token being passed from state i to j. Because the token can only move one step for each horizon, $P_{i,j} = 0$

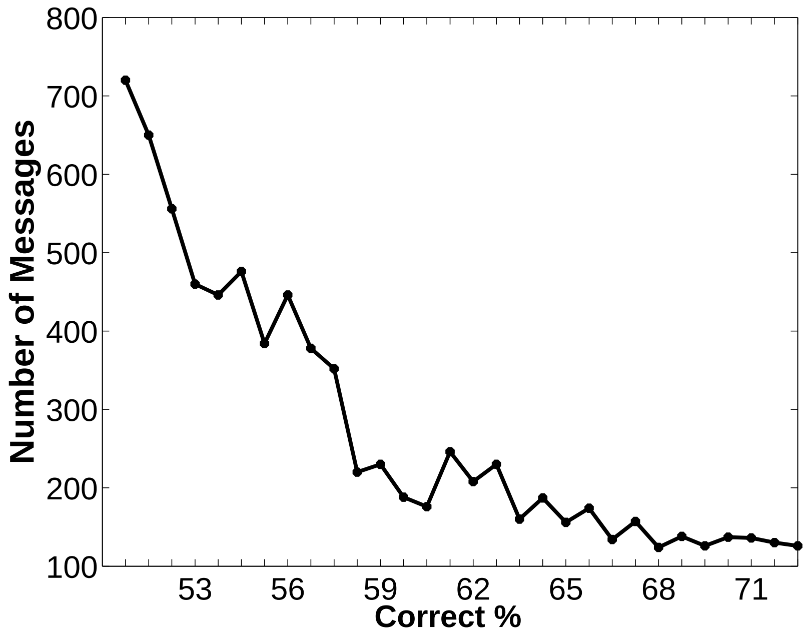


Figure 6: Agents' likelihood of correctness for where to pass a token can dramatically influence the efficiency of coordination in an agent team of 800,000 members.

except for $j \in \{i-1, i, i+1\}$. Therefore, for a state $s_i \neq s_0$, the token may move closer to the destination ($P_{i,i-1}$), stay on the same level ($P_{i,i}$), or move far away ($P_{i,i+1}$). When the token reaches state S_0, it will be stopped at the destination and the probability $P_{0,0} = 1$. For example, suppose u is the initial probability distribution of the token being in state S. According to the theory of Markov chains [57], we can calculate the probability that the token reaches the sink agent after n steps as $P_S^n = u \times P^n$.

For example, suppose that an associate network has only the maximum distance of four andthe token initial probability distribution is $u = [0.60.30.10]$. Furthermore, suppose that the transaction probability matrix P is:

$$\begin{pmatrix} 0.1 & 0.9 & 0 & 0 \\ 0 & 0.4 & 0.6 & 0 \\ 0 & 0 & 0.4 & 0.6 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

After step=10, the probability distribution is [0.00010.00100.00690.9921], which means

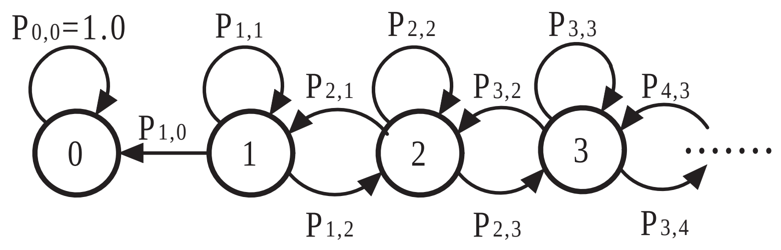


Figure 7: A transition model of tokens' movement based on Markov Chains, where s_i represents the state that the token i leaves far from the destination.

that in more than 99% of cases the token has reached the destination agent.

5.4 TOKEN MOVEMENT OVER DIFFERENT NETWORK TOPOLOGIES

In this section, I investigate the nature of different social networks and their potential influences on the efficiency of token movement. Figures 8 and 9 show the relative rates of $P(s_i, s_{i-1})$ (marked "Close"), $P(s_i, s_{i+1})$ (marked "Further"), and $P(s_i, s_{i+1})$ (marked "Same") for scale-free and random networks. The x-axis shows the distance from a node to the target node, i.e., the subscript i. Note that the closer to the target, the more likely it is that random movement will lead further from the target. Conversely, the further from the target, the more likely it is that random movement will lead the token closer to the target. The figures show that the closer a token is to the target, the easier it is to move the token away. Moreover, since the figures show different distributions, their token movement characteristics are likely to be different.

Figure 8 shows a typical scale-free network. The state probability transition matrix P

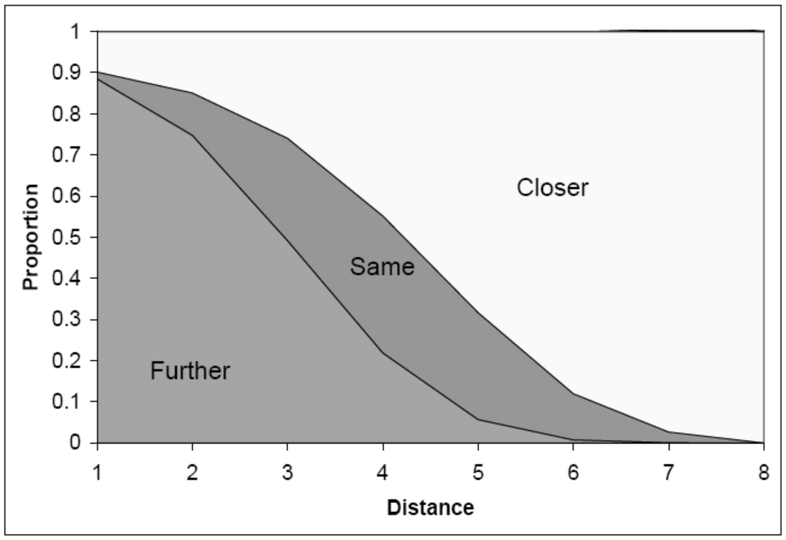


Figure 8: The relative proportions of links in a scale-free network that lead closer to, keep the same distance from, or move further from some target node, as the distance to the target is varied.

in this network topology is:
$$\begin{pmatrix} 0.02 & 0.98 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.01 & 0.1 & 0.89 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0.25 & 0.7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0.35 & 0.45 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.25 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.75 & 0.1 & 0.15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.89 & 0.01 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 9 shows a typical small world network. I can compose the state transition prob-

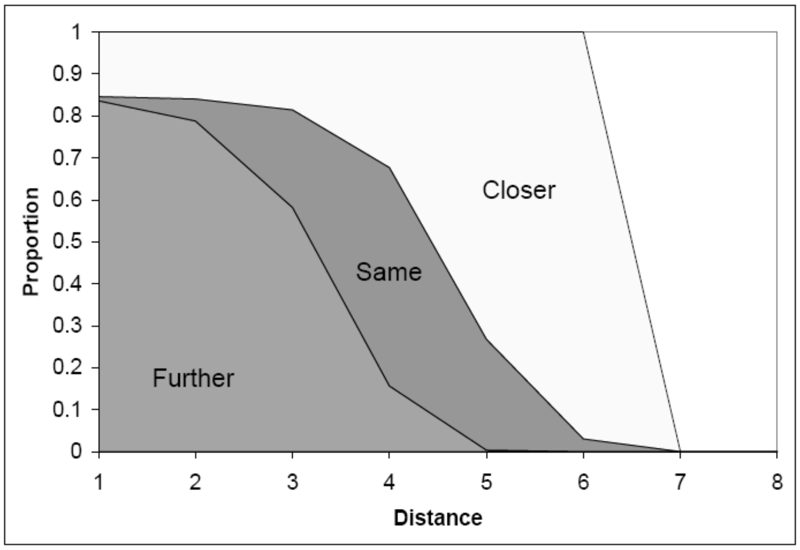


Figure 9: The relative proportions of links in a random network that lead closer to, keep the same distance from, or move further from some target node, as the distance to the target is varied.

ability matrix P as: $\begin{pmatrix} 0.01 & 0.99 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.01 & 0.03 & 0.96 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0.24 & 0.75 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.15 & 0.5 & 0.45 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0.25 & 0.15 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0.05 & 0.15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.88 & 0.02 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

Suppose that the same token's initial distribution is $u = [0.250.250.150.150.10.050.040.01]$ and, after token random movement for 1000 steps, the state probability distribution for a scale-free network is [0.0000 0.0011 0.0194 0.0675 0.0605 0.0198 0.0030 0.8287], where in 83% of cases this token has reached the destination agent. On the other hand, the state probability distribution for a random network is [0.0001 0.0054 0.5405 2.8811 1.9608 0.3378 0.0492 0.1004], where in only about 10% cases this token has reached the destination agent.

5.5 TOKEN MOVEMENTS ON INTELLIGENT AGENT TEAMS

However, in teams, token does not simply move randomly from agent to agent when agents can build a better decision model. Often the agents will know something (perhaps a lot) about the characteristics of their network associates. Several sociologists have shown how information delivery can be very efficient in human teams with simple models of acquaintances[96, 99]. Xu[102] has effectively illustrated this for multi-agent teams.

To model the fact that such movement is not completely random, but is in fact biased towards the target location, we use a parameter β to make $P(s_i, s_{i-1})$ larger and $P(s_i, s_{i+1})$ smaller. However, this bias should be stronger as the token moves nearer to the target location because it is more likely that agents that need the target information know what is required to intelligently route the information. We can model this by using $\beta(i) = \frac{1}{e^{\alpha i}}$. Informally, one can think of β as the total learning of the team about the team state and of α as how much more agents "near" an agent know about it than agents "far" from it do. Using α and β, the Markov Chain state transitions can be rewritten as:

$$\breve{P}(s_i, s_{i-1}) = P(s_i, s_{i-1}) + (1 - e^{\beta(i)})P(s_i, s_i) + (1 - e^{2\beta(i)})P(s_i, s_{i+1})$$

$$\breve{P}(s_i, s_i) = P(s_i, s_i) - (1 - e^{\beta(i)})P(s_i, s_i)$$

$$\breve{P}(s_i, s_{i+1}) = P(s_i, s_{i+1}) - (1 - e^{2\beta(i)})P(s_i, s_{i+1})$$

Figure 10 shows the effect on the scale-free distribution from Figure 8. When they are especially close to the target (i.e., the left of the graph), tokens are much more likely to get closer to the target. Clearly, the result will be the more efficient delivery of tokens when there is a bias as described above.

Based on this graph, we reconstruct the state transaction probability matrix P as:

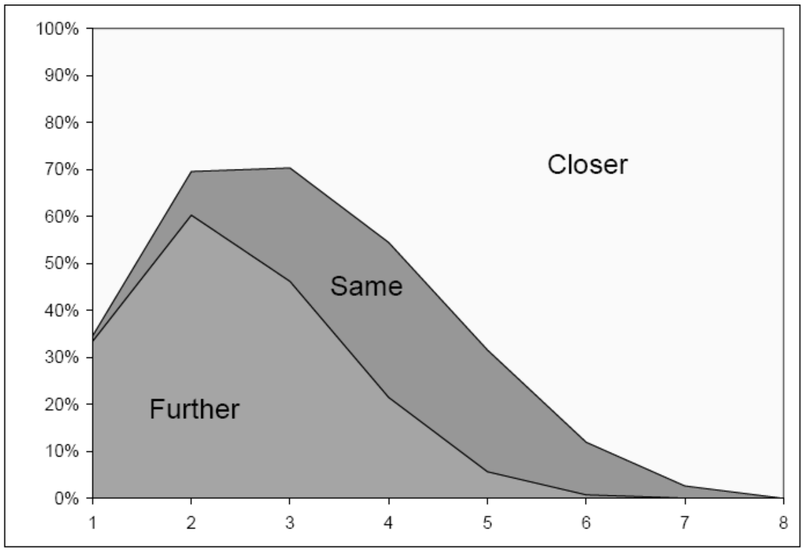


Figure 10: The relative proportions of links that lead closer to, keep the same distance from, or move further from some target, as the distance to the target is varied. In addition, an α value of 1.5 is used to bias the links towards moving towards the target node.

$$\begin{pmatrix} 0.02 & 0.98 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.01 & 0.14 & 0.85 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0.25 & 0.7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0.35 & 0.45 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.45 & 0.25 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.6 & 0.1 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.34 & 0.01 & 0.65 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Compared with the scale-free and random networks in random walk, if a token's initial distribution is the same as in the last section (i.e., $u = [0.250.250.150.150.10.050.040.01]$, after 1000 movements), we will find that the state probability distribution is [0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000] and that the token will surely reach the destination. Moreover, after 100 moves, the state probability distribution is [0.0000 0.0005 0.0078 0.0262 0.0242 0.0094 0.0029 0.9289]. This means that in 92% of cases, the token has reached the destination agent, which is better than a token's random moves over 1000 steps in the same network topology.

5.6 TOKEN MOVEMENT IN THE LOCAL PROBABILITY MODEL

The key to my intelligent routing algorithm is using previously received tokens to train agents' local probability models and agents more likely to forward tokens to an associate who can make use of a token or knows who does. To verify the feasibility of this algorithm, I set up an experiment with an agent team consisting of 400 agents. Each agent has, on average, four associates. One agent is randomly chosen as the source, which a specific target token Δ_i comes from, and another agent is randomly picked as the sink agent where the token will go. Before the movement of the target token Δ_i, the sink agent first sends out 20 tokens (This is called Δ_j) relative to Δ_i to train the team, and each will move TTL=50. Then the source agent sends out Δ_i with $rel(\Delta_i, \Delta_j)$ varied, and I measure how many steps or messages it takes Δ_i to reach the sink agent. In my experiments, four different types of associate network topologies are involved: two-dimensional grid networks, random networks, small world networks, and scale-free networks. The small world network is based on the grid network with 8% of the links randomly changed. The key difference between the random network and the scale-free network is that the former has a "flat" degree distribution and the scale-free network has a power law distribution. Each point on each graph is based on the average of 1000 runs in an abstract simulation environment. Although 1000 runs is a large number of runs on a desktop, the result graphs are not completely smooth because the variance is high. In all of my experiments, the four different types of associate network topologies: random, small world, grid and scale free network, are set up along with the varied parameter test.

5.6.1 Token movement with different relevance

In Figure 11, the average number of steps that it takes to deliver Δ_i is shown as I varied the strength of the relationship between the tokens originally sent out by the sink agent and the

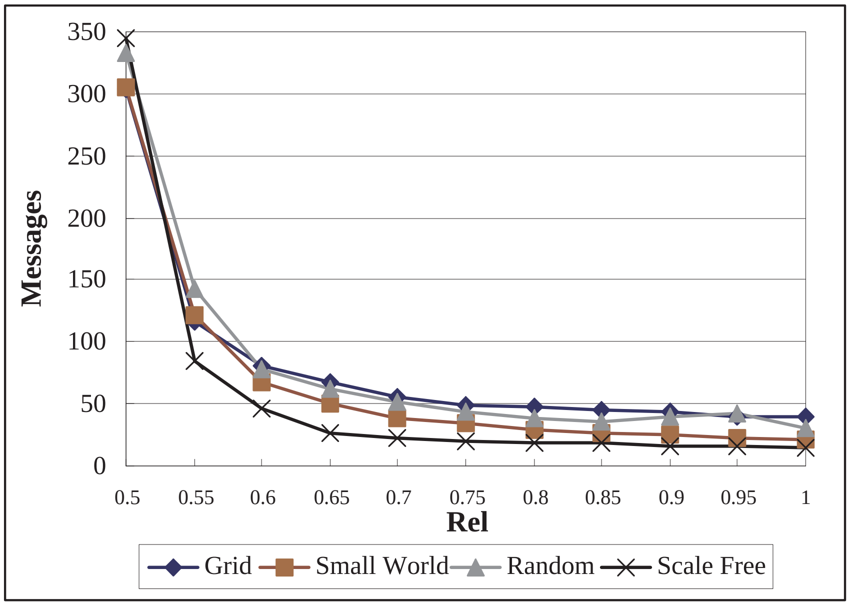


Figure 11: The number of messages decreases dramatically as the association between Δ_j and Δ_i increases.

Δ_i token sent by the source agent between 0.5 to 1. As expected, my algorithm works across all four associate networks. The stronger the relationship between the originally sent token and the Δ_i token, the more efficient the token movement will be. But, very strong relevance does not help token movement much more.

5.6.2 Influence of different numbers of previous tokens

Next, I examine in detail how many messages must be sent by the source to make the delivery from the sink agent efficient. The same settings are used as described above except that the number of messages that the sink agent sends out is varied and the relationship between these messages and Δ_i, $rel(\Delta_i, \Delta_j)$ is set at 0.9. Note that only a few messages are necessary to dramatically impact the number of messages required. This result also shows that a few messages are sufficient for agents to make a "good estimate" of where to send messages. But, overwhelming number of related tokens are not helpful much more.

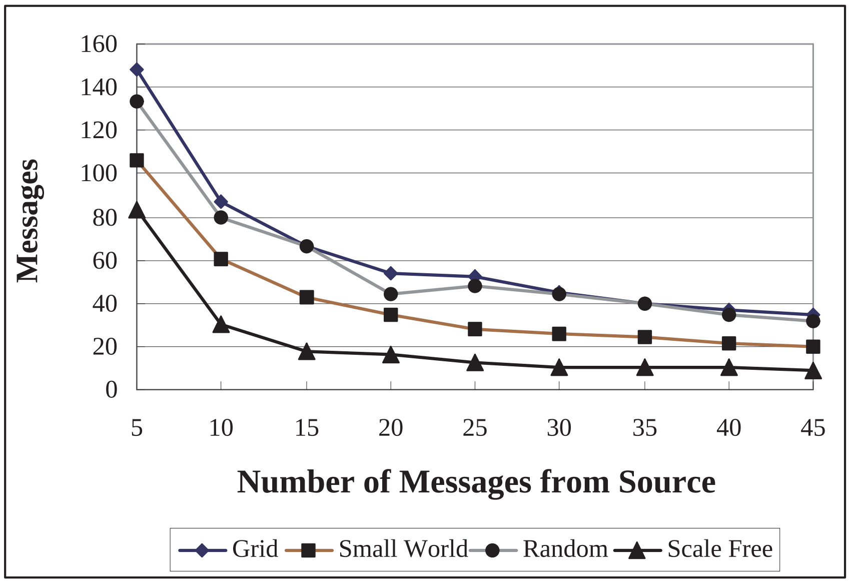


Figure 12: The number of messages decreases as the relative messages from the sink agent increases.

5.6.3 The influence of network density

In the next experiment, I examined how the number of associates can help to make the token-based algorithm more efficient. I run experiments with $rel(\Delta_i, \Delta_j) = 0.8$ and, in associate networks, each agent has an average number of associates that ranges from two to eight. The result in Figure 13 shows that the higher the number of associates, the more messages are made to deliver Δ_i. This demonstrates that token movement cannot be enhanced by connecting an agent with more associates. Moreover, in my experiment, I did not consider the limitation of communication breadth for agent members.

5.6.4 Token movement among different sizes of teams

To investigate the influence of team size on token movement performance, as shown in in Figure 14, I ran experiments using different sizes of teams that ranged from 100 to 550 agents with $rel(\Delta_i, \Delta_j) = 0.7$. Its efficiency is measured as the percentage of agents involved

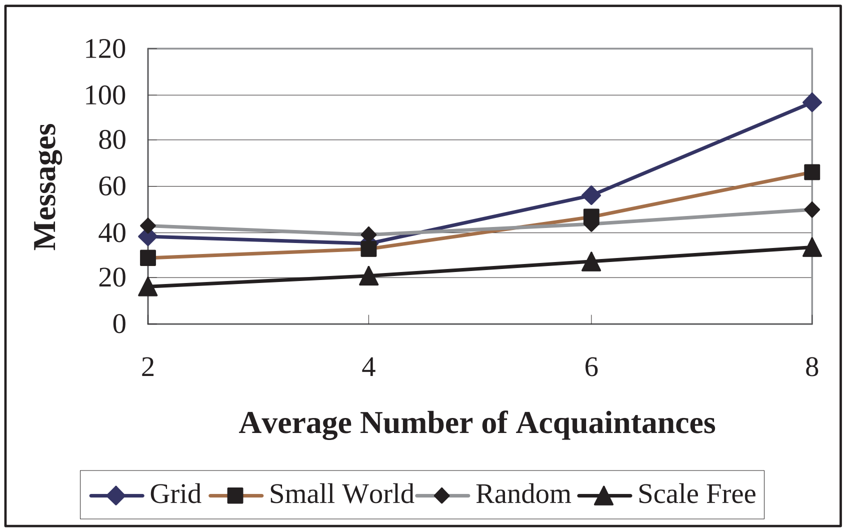


Figure 13: The number of messages increases slightly if each agent has, on average, more associates in associate networks.

in token delivery $percentage = \frac{agents\ involved\ in fodelivery}{Total\ \#\ of\ agent\ team}$. The experiment in Figure 14 shows that, with different sizes of teams, the efficiency of token movement is almost the same. This indicates that team size is not a factor in efficiency.

5.6.5 Token movement with different network structures

From the above experiments, I find that my algorithm works on each type of associate network. I also see clues to how these network topologies influence the efficiency of token movement. I notice that networks with a small average distance [1] (i.e., random, small world, and scale-free networks) always outperform the regular grid network, which does not have such a property. Moreover, a scale-free network with power law distribution is clearly superior to others that do not possess this characteristic. The difference between different associate topologies is distinct when the previous messages have a strong relationship with Δ_i. For example, in Figure 14, when $rel(\Delta_i, \Delta_j) = 1$, the number of messages needed to

[1]The average distance $l = \frac{1}{\frac{1}{2}n(n+1)} \sum_{a_i, a_j \in A, i>j} distance(a_i, a_j)$, where $n = |A|$ and $distance(a_i, a_j)$ represents the minimum number of agents a_i, a_j that a message must pass through in the associate network.

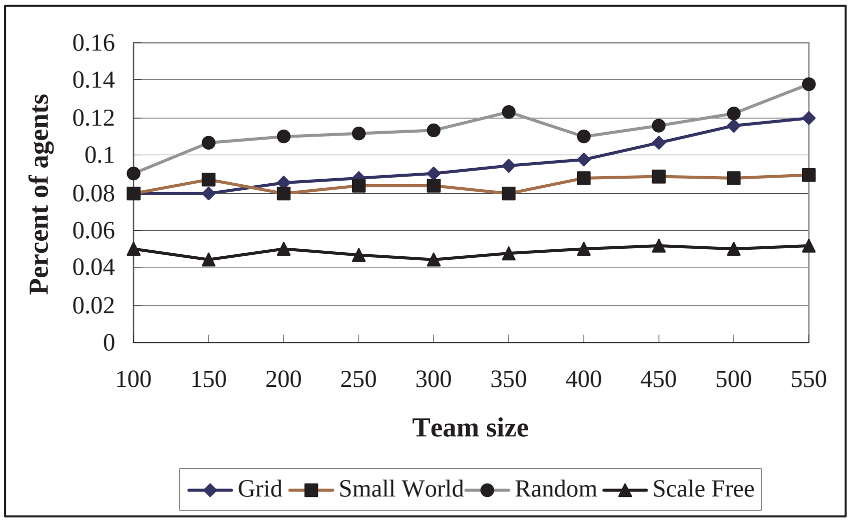


Figure 14: Token movement becomes slightly better for large-scale teams according to the measure of percentages.

deliver Δ_i in a scale-free network is only one-third as many as are required in a grid network.

6.0 DECISION THEORETICAL AGENTS

The major issue we leave for the remainder of this paper is how to design the algorithm for $RouteToken(Next, \Delta_j)$ (described in Section 4.2) if agent a_i does not wish to keep the token. Based on our joint intention model, tokens must be moved to agents who are appropriate recipients of that token. e.g., two events of hearing the fire alarm and seeing smoke in the same building will be delivered to the same agent if one agent can activate the solution of fire fighting; the role token of driving a fire truck role will be allocated to an agent who is capable of driving a fire truck and a resource token of a fire truck will then be passed to that agent.

6.1 MDP MODEL WITH JOINT ACTIVITY

The general model for team coordination is a centralized Markov Decision Process (MDP) with joint activity. It is a tuple: $< A, S, \Theta, T, R >$. A is the team to be coordinated and, for each agent $a_i \in A$, S is the state space and the specific state in time t is $s(t)$. Θ is the joint action space of team A. $T : S \times \Theta \to S$, is the transition function that describes the resulting state $s(t+1) \in S$ when executing $\theta(t) \in \Theta$ in $s(t)$. $R : S \to \mathbb{R}$ defines the instantaneous reward for being in a specific state.

In this case, $s(t)$ is modeled as how the exclusive coordination Ξ is distributed in A:

$$\begin{aligned} s(t) = << Hold(\Delta_1, t), Hold(\Delta_2, t), Hold(\Delta_3, t), ... >, \\ < Know(\Delta_1^I, t), Know(\Delta_2^I, t), Know(\Delta_3^I, t), ... >> \end{aligned}$$

where $TOKEN = \{\Delta_1, \Delta_2, \Delta_3, ...\}$ and $INF = \{\Delta_1^I.tc, \Delta_2^I.tc, \Delta_3^I.tc, ...\}$. $s(t)$ includes two parts:

- $Hold(\Delta_i, t) \in A$ directly describes where a token Δ_i is being held by one of its team members in A at t (e.g., $Hold(\Delta_i, t) = a_j$).
- $Know(\Delta_i^I, t) \subset A$ denotes the information token Δ_i^I that is known by a few members of team A (e.g., $Know(\Delta_i^I) = \{a_i, a_j, a_k\}$).

Since the tokens represent resources, roles, and information, $s(t)$ unambiguously defines who is doing what, with what resources, and with what information. An initial state $s(0)$ denotes the initial team state–for example, that agents on the team have nothing to do and that no environmental event is detected.

Team action Θ is a joint activity in which team members jointly move tokens that they are holding. The joint team action at time-point t is defined as $\theta(t) \in \Theta$, which represents all of the actions that team members are doing: $\theta(t) = \rho_1(t) \wedge \rho_2(t) \wedge ... \wedge \rho_{|A|}(t)$, where $\rho_i(t)$ is agent a_i's action at time-point t. If an agent holds a token, the actions available to an agent a_i is to keep it or to pass it to any other teammate, $\rho_i(t) = \{\rho_i^{a_i}, \rho_i^c | \forall c \in A, c \neq a_i\}$. For convenience, we write $\theta(t)$ as θ and $\rho_i(t)$ as ρ_i when there is no ambiguity.

$R(s(t)) > 0$ when at $s(t)$, and a sub-goal g_i are achieved. The team will be credited with an instant reward valued at $R(s(t)) = \sum_{r \in tis_{i,j}.q_{i,j}} EU(r)$.

The utility of state S under a policy π is defined as

$$v^\pi(s) = \sum_{t=0:\infty} (d^t \times R(s(t)) - t \times commcost)$$

where *commcost* is the communication cost and $d < 1$ is a pre-defined discount factor. $v^*(s)$ allows the agent to select actions according to the optimal policy

$$\pi^*(s(t)) = argmax_{\theta \in \Theta} v^*(s(t+1))$$

By value iteration, $v^*(s(t)) = argmax_{\theta \in \Theta}[R(s(t)) - commcost + d \times v^*(s(t+1))]$. Policy π^* tells the team how to control all of the agents in order to move tokens to maximize the team's expected utility.

We define a matrix V where each element $V[s(t), b] = R'(s(t)) - commcost + d \times v^*(s(t+1))$ when $\chi = move(\Delta, b)$. Then $V[b]$ represents the expected utilities vector for α to send token Δ to b in each different state $s(t)$.

The MDP model with joint activity cannot be applied when the team size greatly increases because agents can neither obtain an exact model of S nor know precisely what the other teammates' intentions are. Moreover, the coordination is decentralized. A team member must coordinate the token with its own knowledge in parallel, which is, in most cases, an incomplete map of $s(t)$ at any given time.

In this section, we have presented three steps of approximations. First, large-team coordination is decentralized and we must approximate the joint activity MDP as an individual activity MDP. Second, we approximate the MDP to a POMDP when the team size greatly increases or agents cannot obtain complete observation of the team. Third, a major assumption is that, within a large-scale team, free communication is not available between any two team members. We approximate full individual activity as partial activity such that agents can only pass tokens directly to a small number of team members.

6.2 DECENTRALIZED MDP FOR TOKEN ROUTING

As the first step, by dividing the monolithic joint activity into a set of actions that individual agents can perform, we can decentralize the token routing process in which distributed agents, in parallel, make independent decisions about where to pass the tokens that they currently hold. Thus, we effectively break a large coordination problem into many smaller ones. Then the MDP model of a single agent a_i making a token routing decision is a tuple: $< a_i, S, \Theta_i, T', R >$. This model can be applied to any other agents on the team.

The major difference between this MDP model and the joint activity model described in the previous section is that Θ is replaced with Θ_i. Θ_i is all of the available individual activities for agent a_i. For each time t, $\rho_i(t) \in \Theta_i$ has been defined in previous sections. Then, in transition function $T' : S \times \Theta_i \rightarrow S$, the team state S will be transmitted according to Θ_i if other than Θ.

Now the individual optimal policy $\pi^{**}(s(t))$ is defined as:

$$\pi^{**}(s(t)) = argmax_{\rho_i \in \Theta_i} v_I^*(s(t+1))$$

. By value iteration,

$$v_I^*(s(t)) = argmax_{\rho_i \in \Theta_i}[R(s(t)) - commcost + d \times v_I^*(s(t+1))]$$

$v_I^*(s(t))$ is the number of rewards according to an individual optimal policy π^{**} and will tell agent a_i to choose an action that carries maximum expected rewards for the global state of the team. In practice, we always choose a relatively small value for d. Although an agent in this situation can only choose an optimal action in states that are close to those that carry instant rewards, many states carry few expected rewards. This will be very helpful for the MDP in terms of quickly finding the optimal activity by only considering a few potential subsequent states with prominent expected rewards. This is very important because, in large-scale team coordination, the number of the candidates in subsequent states is extremely high.

6.3 PARTIAL OBSERVABLE MARKOV DECISION PROCESS

Token-based coordination is a process by which agents attempt to maximize the overall team reward by moving tokens around the team. If an agent were to know the exact state of the team, it could use an MDP to determine the expected utility-maximizing way to move tokens. Unfortunately, it is infeasible for an agent to know the complete state of the team when it is large or in an ad-hoc environment [67]. But it is illustrative to look at how tokens can be passed if it were feasible. In this section, we have presented the second step of approximation in which agents do not have a complete map of the team state $s(t)$ and must make their decisions according to its local state. This model is defined as $< S, L, \Theta_i, Z, O, T', R >$.

L maintains the local model of agent a_i and, at each time-point t, it is defined as $l_{a_i}(t) =< tokens(a_i), h_{a_i}(t) >$, where $h_{a_i}(t)$ includes all of the tokens that agent a_i has previously passed if $a_i \in \Delta.path$, $\Delta \in h_{a_i}$. As defined in Section 3, $tokens(a_i)$ are all of the tokens currently held by a_i.

The observation function is $O : L \rightarrow Z$, where Z maintains a local observation from L. Each observation at time t is $z_{a_i}(t)$ and includes two parts:

$$
\begin{aligned}
&z_{a_i}(t) = \\
&<< PrevHold(\Delta_1, \Delta_1.path), PrevHold(\Delta_2, \Delta_2.path), \ldots, >, \\
&< PrevKnow(\Delta_1^I, \Delta_1^I.path), PrevKnow(\Delta_2^I, \Delta_2^I.path), \ldots >>
\end{aligned}
$$

In the first part, $\forall \Delta_j \in h_{a_i}(t)$, $PrevHold(\Delta_j, \Delta_j.path)$ denotes that a_i has observed that all of the agents in $\Delta_j.path$ have previously held Δ_j. In the second part, $\forall \Delta_j^I$, $\Delta_j^I.tc \in INF$ and $PrevKnow(\Delta_j^I, \Delta_j^I.path)$ denote that the encapsulated information Δ_j^I has been known by the agents that Δ_j^I has previously reached.

We adopt a standard POMDP techniques called Q-MDP [50, 52] and use this technique to solve the POMDP and thus determine optimal token routing. In this solution, a_i's individual belief $b_{a_i}(t)$ is defined as a set of possible team states $b_{a_i}(t) \subseteq S$. This denotes that the agent a_i believes that the previously possible team state based directly on its observation $z_{a_i}(t)$.

The mapping function defined as

$$z_{a_i}(t) \rightarrow b_{a_i}(t)$$

is a peer-to-peer function. It will exclude all the $s(t) \in S$ that are compatible with $z_{a_i}(t)$. For example, a_i observes that Δ_j is held by a_i. All of the states that denote a_i are not holding Δ_j and will be excluded.

For each state $s(t) \in b_{a_i}(t)$, we supposed, for each agent a_i, that we know the perceptual distribution $Pr(s(t)|z_{a_i}(t))$, which describes the likelihood that the team is in the state $s(t)$ when its observation is $z_{a_i}(t)$ and $\sum_{s(t)\in b_{a_i}} Pr(s(t)|z_{a_i}(t)) = 1$. Initially, we have $Pr(z_{a_i}(t)|s(t)) = \frac{1}{|b_{a_i}(t)|}$ when none of the $s(t) \in b_{a_i}(t)$ is prominent. Then we can calculate the expected reward of each observation $z_{a_i}(t)$ as:

$$\begin{aligned}R'(z_{a_i}(t)) &= R'(b_{a_i}(t)) \\ &= \sum_{s(t)\in b_{a_i}} Pr(s(t)) \times (v_I^*(s(t))) \\ &= \sum_{s(t)\in b_{a_i}} Pr(s(t)|z_{a_i}(t)) \times (v_I^*(s(t)))\end{aligned}$$

Although the transition function T' is the same as the previous MDP for a decentralized models, the local policy under POMDP π_P^* is to select the action Θ_i:

$$\begin{aligned}&argmax_{\rho_{a_i}(t)} R'(z_{a_i}(t+1)) \\ &= argmax_{\rho_{a_i}(t)} R'(b_{a_i}(t+1)) \\ &= argmax_{\rho_{a_i}(t)} \sum_{s(t+1)\in b_{a_i}(t+1)} Pr(s(t+1)|z_{a_i}(t+1)) \times v_I^*(s(t+1))\end{aligned}$$

This formula is based on the assumption that we can learn online or off-line from the policy π^{**} in the previous decentralized MDP model that the optimal team reward $v_I^*(s(t))$ for each state $s(t)$ and Q-MDP will always choose to pass a token to the team member who is the most likely maximize the team rewards from agent a_i's local observation.

6.4 POMDP BASED ON ASSOCIATE NETWORK

Although the Q-MDP approach in theory is computable and can solve our POMDP problem, a major difficulty for system design is that Q-MDP must compute every teammate as a potential recipient for every incoming token. Moreover, in some application domains where teams work in hostile environments, free communication is not available to any pair of agents. One solution is to limit the actions needed to pass a token to only a few teammates. Therefore, tokens are required to move around the associate network.

We rewrite this POMDP model as $< S, L, \Theta_i^*, Z, O, T'', R >$ where $T'' : S \times \Theta_i^* \rightarrow S$. But the major difference is that the available action $\rho_{a_i}^*(t) \in \Theta_i^*$ for each agent a_i is changed to keeping or passing it to one of its associates as

$$\rho_{a_i}^*(t) = \{\rho_i^{a_i}, \rho_i^c | \forall c \in n(a_i)\}$$

Then the optimal policy π_I^{**} under this associate network is:

$$\begin{aligned} & argmax_{\rho_{a_i}^*(t)} R'(z_{a_i}(t+1)) \\ &= argmax_{\rho_{a_i}^*(t)} \sum_{s(t+1) \in b_{a_i}(t+1)} Pr(s(t+1)|z_{a_i}(t+1)) \times (v_I^*(s(t+1))) \end{aligned}$$

From the formula above, we can find the three things that determine the reward for passing a token to an associate: agent a's observation, the probability distribution of the current state, and the expected reward for that team state.

6.5 APPROXIMATING THE THEORETIC MODEL AS A PROBABILITY MODEL

Following the idea of Q-MDP based on associate network, agent a makes use of V if we re-design observation function as $O : Z_a \times S \rightarrow \Omega_a$. *Belief state* Ω_a is a discrete probability distribution vector over the team state $s(t)$ inferred from the current local state z_a. For example, if $S = \{s_1,\ s_2,\ s_3\}$ and $\Omega_a = [0.6, 0.2, 0.2]$, a estimates that the probability of $s(t)$ being s_1 is 0.6 and being s_2 and s_3 are 0.2.

To calculate the expected utility vector, $EU(\Omega_a) = \Omega_a \times V$. For example, if a has associate b, c, and d and $EU(\Omega_a) = [5, 10, 6, 4]$, then $EU(\Omega_a, b) = 10$ represents the expected utility to send Δ to b according to the Q-MDP. The locally optimal policy $\pi^{**}(\Omega_a)$ is $argmax_{\chi \in Action_a} EU(\Omega_a, c)$. This is the action that an agent should take in order to maximize expected utility, given that the agent has an incomplete view of the team state. As in the previous example, passing Δ to b is the best choice because $EU(\Omega_a, b) = 10$ is the maximum value of $EU(\Omega_a)$.

If the local policy π^{***} always matches π^{**}, $P_a[\Delta]$ will be the normalization of $EU(\Omega_a)$.

$$\forall b \in n(a), \quad P_a[\Delta, b] = \frac{EU(\Omega_a, b)}{\sum\limits_{c \in N(a)} EU(\Omega_a, c)}$$

That is, the largest expected utility for sending a token to an associate should result in the highest probability. Following the previous example, if $EU(\Omega_a) = [5, 10, 6, 4]$, then for optimal behavior $P_a[\Delta] = [0.2, 0.4, 0.24, 0.16]$.

Now we are in a position to see how the receipt of a token affects the locally optimal policy for routing token Δ and thus to determine how to compute relevance. Suppose that the state estimation of agent a just before a token Δ_{pre} arrives is Ω_a and, after it arrives, the state estimation changes to Ω'_a because a gains additional knowledge from the newly received token. According to Q-MDP, before Δ_{pre} is received, the expected reward of a is $EU(\Omega_a) = \Omega_a \times V$ while, after the arrival of Δ_{pre}, $EU(\Omega'_a) = \Omega'_a \times V$. Moreover, agent a's local model will also be updated according to Δ_{pre}. Suppose Δ_{pre} comes from associate b and $P_a[\Delta, b]$ is the probability that a will send Δ to b before Δ_{pre} comes, while $P'_a[\Delta, b]$ is the updated probability after the arrival of Δ_{pre}. According to our assumption that the policy π^{***} (according to the P_a model) and π^{**} (according to the POMDP model), the agent will choose the same action, which is to send Δ to b. Thus, we have

$$\frac{P'_a[\Delta, b]}{P_a[\Delta, b]} = \frac{EU({\Omega'}_a, b)}{EU(\Omega_a, b)} = \frac{[\Omega'_a \times V]_b}{[\Omega_a \times V]_b}$$

where $[\Omega'_a \times V]_b$ is the value of the component in vector of $[\Omega'_a \times V]$ according to associate b. It is the same vector as $EU(\Omega'_a, b)$.

We can conclude that a received token changes the agent's estimation of the probability distribution of the team's state, which in turn directly influences the decision of where to send related tokens. If we know a little bit of how the probability distribution changes for an agent after it has passed a token, we can use this to predict how this agent updates its distributed decision model; therefore, we can define the relevance between tokens. A heuristic that captures this relationship will approximate the locally optimal policy and hence lead to good performance.

7.0 ROUTING TOKENS

In this section, I provide a heuristic approach for token-based team coordination inspired by the local POMDP. This approach yields fast, efficient routing decisions without requiring accurate knowledge of the complete team state.

7.1 LOCAL HEURISTIC APPROACH

Token-based coordination is a process by which agents attempt to maximize the overall team reward by moving tokens around the team. If an agent were to know the exact state of the team, it could use an MDP to determine the expected utility-maximizing way to move tokens. Unfortunately, it is infeasible for an agent to know the complete state. However, in [67], it is illustrative to consider how tokens would be passed if it were feasible. By dividing the monolithic joint activity into a set of actions that can be allocated to individual agents, we can decentralize the token routing process where distributed agents, in parallel, make independent decisions about where to pass the tokens that they currently hold. Thus, we effectively break a large coordination problem into many smaller ones.

As explained in the last chapter, knowing the complete team state is only feasible for small teams. In large teams, agents must make token coordination decisions based on a more limited view of the team. Thus, the reasoning must be modeled as a Partially Observable Markov Decision Process (POMDP). Standard POMDP techniques, such as [50] and [52], could be used to solve the POMDP to determine optimal token routing. However, for faster routing of tokens, this local POMDP tells the agent the optimal action but the computational complexity is still too high for practical applications. From the POMDP model, we can gain

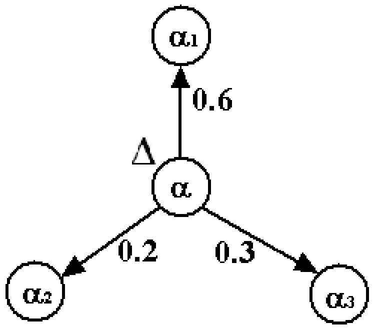


Figure 15: Agent a's local model for where to send Δ. The probability that a_1 is the best to send Δ to is 0.6 and a will pass Δ to a_1 according to π^{***}

important hints about how to take a heuristic approach and build a local decision model.

Following the definition in section 6.5, P_a is the decision that matrix agent a uses to decide where to move tokens. Each row $P_a[\Delta]$ in P_a represents a vector that determines the decision where to pass a token Δ to one of its associates. Specifically, each value $P_a[\Delta, b] \rightarrow [0,1], b \in n(a)$ represents a's decision that the probability of passing token Δ to an associate b would be the action that maximizes team reward. Our policy π^{***} for this local model is to choose action χ to $argmax_{\chi \in Action_a} P_a[\Delta, c]$ where $\chi = move(\Delta, c)$. Figure 15 shows an example where $P_a[\Delta] = [0.6, 0.1, 0.3]$ and agent a has three associates a_1, a_2, a_3. $P_a[\Delta, a_1] = 0.6$, $P_a[\Delta, a_2] = 0.1$, $P_a[\Delta, a_3] = 0.3$, and π^{***} will choose the action $move(\Delta, a_1)$ to pass Δ to a_1. The key to this distributed reasoning lies in how the probability model P_a for each agent a is updated. If the action indicated by P_a matches the optimal policy π^* from the MDP model, then the team will act optimally.

Initially, agents do not know where to send tokens, but as tokens are received, a model can be developed and better routing decisions can be made. That is, the model P_a is based on accumulated information provided by the receipt of previous tokens. For example, when an agent sends a role token to an associate that has previously rejected a similar role, the team is potentially hurt because this associate is likely to reject this role as well. In such a case, communication bandwidth has been unnecessarily wasted.

From this perspective, P_a can only depend on a's history of received tokens, H_a. The update function $Update(P_a[\Delta], \Delta_i)$ for $P_a[\Delta]$, defines the calculation of the probability vector for where to send Δ based on previously received tokens Δ_i in H_a. This will be explained in detail in the next sections.

Algorithm 2 shows the reasoning of agent a when it receives incoming tokens from its associates via the function $getToken(sender)$ (line 2). For each incoming token Δ, the function $Acceptable(a, \Delta)$ determines whether the token will be kept by a (line 4). When a resource token is kept, its threshold is raised (line 6). If a decides to pass Δ, it will add itself to the path of Δ (line 9) and $Update(P_a[\Delta], \Delta_i)$, will update how to send Δ according to each previously received token Δ_i in a's history (line 11). If Δ is a resource or role token, its threshold will be decreased (line 14). Then a will choose the best associate to pass the token to according to $P_a[\Delta]$ (line 16) and record Δ in its history, H_a (line 18).

Algorithm 2: Decision process for agent a to pass incoming tokens.

```
1:  while true do
2:    $Tokens(a) \leftarrow getToken(sender)$;
3:    for all $\Delta \in Tokens(a)$ do
4:      if $Acceptable(a, \Delta)$ then
5:        if $\Delta.type == Res$ then
6:          $Increase(\Delta.threshold)$;
7:        end if
8:      else
9:        $Append(self,\ \Delta.path)$;
10:       for all $\Delta_i \in H_a$ do
11:         $Update(P_a[\Delta], \Delta_i)$;
12:       end for
13:       if $(\Delta.type == Res)||(\Delta.type == Role)$ then
14:         $Decrease(\Delta.threshold)$;
15:       end if
16:       $associate \leftarrow Choose(P_a[\Delta])$
```

```
17:         $Send(associate, \Delta)$;
18:         $AddtoHistory(\Delta)$;
19:       end if
20:    end for
21: end while
```

7.2 INFLUENCE DIAGRAM OF TOKEN-BASED APPROACH

In this section, I investigate the local decision model from the perspective of each individual agent. As explained in Chapter 6, to make the optimal decision, agent has to clear know the team state in each time. On the other hand, team states have been encapsulated into the tokens. Any changes of environment when detected will be encapsulated into information token while any newly created role will be encapsulated into role tokens. When a role token is accepted, its movement will be stopped. Therefore, any changes of team states will directly influent the distribution and movement of tokens. If an agents knows all the distributions of the tokens, it will know the exact team state. In this view point, tokens can be deemed as a direct projection of the team state. The idea of our token-based coordination is making use of the token movement to support agents coordination decision. The influence diagram of how an agent makes a decision is shown in 16.

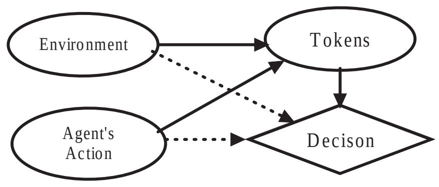


Figure 16: Influence diagram for a localized decision model in the token-based approach.

In this model, an agent's decision-making process can be understood as composed of two parts: its environment and its own activities. On the other hand, a token's movement are also based on these two parts. The coordination decision is to choose the best action either move or keep a token to maximize the expected rewards function. The computation of utility function requires take the complete team states into consideration. In token-based coordination, the computation of the utility function will be based on the tokens which is the direct projection of the team state. An advantage of the token-based coordination is that it successfully factorized the team state. Each piece of token represents a factor of the team state. Team state has to be mathematically expressed as a whole entity and computer is hard to be taught which part of the team states will influence the decision of current activities. When factorized into tokens, expert in domain is easily to tell the causal relationship between the tokens. In my approach, because of the incapability to observe all events in the environment, agents use all of their incoming tokens to make decisions.

7.3 LOCALIZED DECISION MODEL

Suppose $T(t)$ is all the tokens around the team in and before time t, then for any agent α the probability of taking an action a for all its available activities σ can be calculated as:

$$U(a|s(t)) = U(a|T(t))$$

$$= \sum_{i \in RESULTs_a} Pr(result_a^i | a, T(t)) \times U(result_i)$$

$$= \sum_{i \in RESULTs_a} Pr(result_a^i | a, T_i') \times U(result_i)$$

Where $T_i' \subseteq T(t)$ is only a part of tokens which have causal relationship between $result_i$ if action a is token. From the equation above, the utility of rewards for agent α taking an action a depends on the likelihoods of the results are and what is the rewards if the results comes into true.

By utilizing tokens, influence diagram algorithm factorizes team state to support decision. But it still requires the complete team states or token distribution which is unfortunately infeasible for large team coordination. For an agent α, all the tokens it knows is defined as

$T_\alpha(t)$. For supporting the decision to choose an action a, if $U_{i \in RESULTs_a} T'_i \subseteq T_\alpha(t)$, α will choose the optimal action. If we can route all the tokens for a specific decision to α, it will act optimal, moreover, current token will be used to route the rest token in turn. Although we tried to gather all the related tokens to a specific agent, we cannot guarantee. Agents try to find which neighbor is potentially best to passing a token. This probability is based on the expected rewards to pass the token to each neighbor based on the incomplete information. Therefore,

$Pr(a|s(t)) = Pr(a|T_a(t)) = U(a|T_a(t)) / \sum_{a' \in \sigma} U(a'|T_a(t))$

If we do not have any preference for any results, $U(result_i)$ will be normalized. Or, we can simply add an importance factor β. Then, $Pr(a|s(t))$ only depends on $Pr(result^i_a|a, T'_i)$.

Suppose $T'_i = t_1, t_2, ...t_n$,

$$Pr(result^i_a|a, T'_i = Pr(result^i_a|a, t_1 \cup t_2, ... \cup t_n)$$

$$= Pr(t_1 \cup t_2, ... \cup t_n|a, result^i_a) \times Pr(a|result^i_a)/Pr(t_1 \cup t_2, ... \cup t_n|a)$$

$Pr(t_1 \cup t_2, ... \cup t_n|a) = Pr(t_1 \cup t_2, ... \cup t_n)$ is a constant because T'_i independent with a. Therefore,

$$Pr(result^i_a|a, t_1 \cup t_2, ... \cup t_n)$$

$$= \delta \times (\sum_{j=1:n} Pr(result^i_a|a, t_j))$$

We can see that if we know the probability to get a desired results by send current token at the evidence of currently existing token, agent can make rational decision.

Although $Pr(result^i_a|a, t_j)$ is hard to be calculated, we can defined it with a rational value in domain. When a token is received, it will be added as an extra evidence how the optimal action is, which is to send another related token.

7.4 RELEVANCES BETWEEN TOKENS

As explained in Section 6.4, the object of a decision is $argmax_{\chi \in Action_a} EU(\Omega_a, b)$, where b is one of agent a's associates. In my localized decision model, this function is no longer a function related to the entire team state or the external environment, but a function of all of the history of previously received tokens $H(a)$. Clearly, agent a makes decision based on $P'_a[\Delta, b]$, which is the probability of sending token Δ to b. From Section 6.5, we have: $\frac{P'_a[\Delta,b]}{P_a[\Delta,b]} = \frac{EU(\Omega'_a,b)}{EU(\Omega_a,b)}$

Now we can approximate $\frac{P'_a[\Delta,b]}{P_a[\Delta,b]} \approx \frac{EU(H'_a,b)}{EU(H_a,b)}$. Where $H'_a = H_a + \Delta$, I write $H_a = \{\Delta_1, \Delta_2, ..., \Delta_i, ...\}$. Not all of the tokens are related to a reward. For example, the information token "I am hungry" and a resource token "pizza" are related to feeding a hungry agent, while a role token "editing a thesis" is related to the resource token "laptop" to achieve a goal. But we cannot find the relevance between "pizza" and "computer." To simplify, I assume in H_a that only one token Δ_i is related to Δ. Therefore, $EU(H'_a, b) = EU((\Delta, \Delta_i), b) + EU(H_a, b)$ and $\frac{P'_a[\Delta,b]}{P_a[\Delta,b]} \approx \frac{EU(\{\Delta,\Delta_i\},b)+EU(H_a,b)}{EU(H_a,b)}$.

Since $EU(H_a, b)$ is a constant in this case, $\frac{P'_a[\Delta,b]}{P_a[\Delta,b]}$ only depends on $EU(\{\Delta, \Delta_i\}, b)$, i.e., the relationship between Δ and Δ_i. We refer to this as *relevance*. Deciding where to send one token based on the receipt of another relies on knowing something about the relationship between the two tokens. We quantify this relationship as the *Relevance* and define the relationship between tokens Δ_i and Δ_j as $Rel(\Delta_i, \Delta_j) \approx \frac{EU((\{\Delta_j,\Delta_i\},b)+EU(H_a,b))}{EU(H_a,b)}$. When $EU(\{\Delta_j, \Delta_i\}, b) > 0$ and $Rel(\Delta_i, \Delta_j) > 1$ indicate that an agent with use for Δ_i will often also have use for Δ_j, $EU((\Delta_j, \Delta_i), b) < 0$ and $Rel(\Delta_i, \Delta_j) < 1$ indicate that an agent Δ_i also has use for it but that it is unlikely to have use for Δ_j. If $EU(\{\Delta_j, \Delta_i\}, b) = 0$, and $Rel(\Delta_i, \Delta_j) = 1$, then nothing can be inferred. Details about how *relevance* is computed to ensure appropriate behavior will be explained in the next section.

7.5 INTELLIGENT TOKEN ROUTING ALGORITHM

The effectiveness of the token-based approach depends on how well agents maintain their local models so that tokens are routed to where they lead to the highest gain in expected rewards. In this section, we describe an algorithm to update the localized decision model by utilizing previously received tokens. The key is to make use of the relationships between tokens.

The update function of $P_a[\Delta_j]$ according to H_a, written as $Update(P_a[\Delta_j], \Delta_i)$, where $\Delta_i \in H_a$, is found by using Bayes' Rule as follows:

$$\forall b \in n(a),\ \forall \Delta_i \in H_a,\ d = first(n(a), \Delta_i.path)$$

$$Update(P_a[\Delta_j, b], \Delta_i) = \begin{cases} P_a[\Delta_j, b] \times Rel(\Delta_i, \Delta_j) & \text{if } \Delta_i \neq \Delta_j, b = d \\ P_a[\Delta_j, b] & \text{if } \Delta_i \neq \Delta_j, b \neq d \\ P_a[\Delta_j, b] \times \varepsilon & \text{if } \Delta_i = \Delta_j, b \in \Delta_j.path \cap n(a) \end{cases}$$

Where $Update(P_a[\Delta_j, b], \Delta_i)$ updates the $P_a[\Delta_j, b]$ in $P_a[\Delta_j]$ according to Δ_i and $first(n(a), \Delta_i.path)$ extracts from the recorded path of the token the associate of agent a that had the token Δ_i earliest. The first case in this function is the most important. The probability that the sender of the previous token Δ_i is the best agent to receive the token Δ_j is updated according to $Rel(\Delta_i, \Delta_j)$. The second case in the equation changes the probability of sending that token to agents other than the sender in a way that ensures that the subsequent normalization has the desired effect. Finally, the third case encodes the idea that a should typically not pass a token back to the agent that sent it. $P_a[\Delta_j]$ is subsequently normalized to ensure that $\sum_{b \in n(a)} P_a[\Delta_j, b] = 1$.

To see how the updating function works, consider the following example. Suppose agent a has five associates $\{a, b, c, d, e\}$ and $P_a[\Delta_j] = [0.1, 0.4, 0.2, 0.2, 0.1]$. Moreover, $H_a = \{\Delta_i, \Delta_k\}$, $rel(\Delta_i, \Delta_j) = 1.2$ and $rel(\Delta_k, \Delta_j) = 0.4$. $\Delta_i.path = \{b, ..\}$; $\Delta_k.path = \{c, ..\}$; $\Delta_j.path = \{e, ..\}$. If a currently holds Δ_j, by applying our updating function to $P_a[\Delta_j]$, we find the result $P_a[\Delta_j] = [0.12, 0.56, 0.09, 0.23, \varepsilon]$ and Δ_j will be most likely passed to associate b.

7.6 DEFINING TOKEN RELEVANCE: SIMILARITY

When an agent receives two tokens that are relevant to one another, they are more likely to be usable in concert to obtain a reward for the team. According to the update function $Update(P_a[\Delta, b], \Delta_{pre})$, we should get $P'_a[\Delta, b] = Rel(\Delta, \Delta_{pre}) \times P_a[\Delta, b]$. Thus, the relationship between *Rel* and our POMDP model is:

$$Rel(\Delta, \Delta_{pre}) = \frac{[\Omega'_a \times V]_b}{[\Omega_a \times V]_b}$$

. Therefore, while it is infeasibly complex, the POMDP model can suggest how relevance should be defined. In this paper, we simply estimate this value based on the similarity between tokens. Intuitively, if two tokens are similar, receiving one token allows an agent to update its estimation of the team state and infer where to pass similar tokens. For example, receiving a role token from a particular associate tells the agent that it is relatively less likely that similar role tokens will be accepted in the part of the network accessible via that associate. Receiving an information token with information about Pittsburgh tells the agent that some agents in that part of the network must currently be in Pittsburgh.

The similarities between tokens come from the *coordination* they carry, and the calculation depends on the domain knowledge of the applications. We assume that from $\Delta_i.coordination$ and $\Delta_j.coordination$, we can deduce the similarity between two tokens as $sim(\Delta_i, \Delta_j)$. $sim(\Delta_i, \Delta_j) > 1$ if Δ_i and Δ_j are a pair of similar tokens. For example, if two tokens both reference Pittsburgh, we consider them similar because both are involved with the same location. In the same way, we consider two tokens that require driving a specific machine as similar because they need the same kind of capacity. Two tokens that are both pre-conditions for the same plan would also be considered similar.

We distinguish the relationship between the relevance and similarity of two tokens as positively or negatively related. For two similar tokens Δ_i and Δ_j, if an agent previously received a token from an associate and would prefer to send a similar token to that associate, similar tokens are positively related to each other and $Rel(\Delta_i, \Delta_j) = sim(\Delta_i, \Delta_j)$. Otherwise, if this agent is less likely to send a similar token to that associate, similar tokens are negatively related to each other, so $Rel(\Delta_i, \Delta_j) = \frac{1}{sim(\Delta_i, \Delta_j)}$.

The similarity between different types of tokens potentially influences agents' estimation in different ways. As we have shown in previous examples, the receipt of role tokens discourages sending similar tokens to agents along the role tokens' paths because the previous token senders refused the role token and are incapable of accepting the role; therefore, they are less likely to be interested in the information, tasks, or resources that similar tokens carry. Consequently, a previous role token is negatively related to similar tokens. Similarly, the receipt of an information token will indicate that agents along the information tokens' paths are more likely to work on tasks related to that information and are interested in other similar tokens. Hence, a previous information token is positively related to similar tokens.

If the *threshold* of a resource token Δ_i is greater than its initial value ($init$) upon arrival to the current agent, this means that the resource has been used by the agents previously holding Δ_i and that those agents are potentially engaged in tasks requiring the resource. Therefore, if the current agent receives similar tokens, it will be more likely to send them to the part of the network where the previous token has been passed. In this case, the previous resource token is positively related to similar tokens. Alternatively, if $\Delta_i.threshold$ is lower than its initial value ($init$), it means that agents passing the token did not need it. In such a case, the previous resource token is negatively related to similar tokens.

Suppose Δ_i is a previously received token. Now we can summarize the calculation of $Rel(\Delta_i, \Delta_j)$ according to $sim(\Delta_i, \Delta_j)$. No matter what $\Delta_j.Type$ is, this function only depends on the type of the previously incoming token:

$$Rel(\Delta_i, \Delta_j) = \begin{cases} sim(\Delta_i, \Delta_j) & \text{if } \Delta_i.Type = inf \\ sim(\Delta_i, \Delta_j) & \text{if } \Delta_i.Type = res,\ \Delta_i.threshold > init \\ \frac{1}{sim(\Delta_i, \Delta_j)} & \text{if } \Delta_i.Type = role \\ \frac{1}{sim(\Delta_i, \Delta_j)} & \text{if } \Delta_i.Type = res,\ \Delta_i.threshold < init \end{cases}$$

8.0 RELATED TOPICS

Although I have systematically explained my design of a team-scalable coordination approach based on tokens, many open questions remain. Some of them are critical for the feasibility or efficiency of the multi-agent teams or are important when those systems are supervised by human operators. In this chapter, I attempt to resolve two major open questions.

First, when the team is coordinated on the basis of agents' local imprecise decision models, conflicts or duplications of beliefs are inevitable. Therefore, duplicate or contradictory plans are created. How can I solve the most potential plan conflicts to minimize their influences on coordination efficiency?

Second, to enable adjusted autonomy [82], an accurate view of the multi-agent team's overall properties, such as the percentages of team members that are busy or the overall level of remaining fuel of a UAV team, are required. I will explain how the high-level team status monitoring is feasible with limited communication overhead.

8.1 PLAN DE-CONFLICTION

In this section, I describe how to resolve plan conflicts. When using distributed plan creation, two problems may occur. Upon detecting the appropriate pre-conditions, different team members may create identical plans or plans with the same p_g but different p_{recipe}. For example, suppose a team oriented plan has two pre-conditions, a and b, and one agent received a first and b later, while another agent received b and a at the same time. Two agents are eligible to initiate the plan, but only one is required for the team. We have to handle the potential conflicts of knowledge, plans, and activities of a team based on agents' incomplete knowledge. To reduce the need for plan de-confliction, we need to choose a rule for plan instantiation to reduce the number of plans created with the same p_g. These instantiation rules include *always instantiate, probabilistic* and *local information* [49]. The choice of the plan instantiation rule will vary with the domain setting.

If two plans, $plan_i$ and $plan_j$, have some conflict or potential synergy, then we require $sub-team_i \cap sub-team_j \neq \emptyset$ to detect it. There must be a common team member on both sub-teams to maintain mutual beliefs of the plans and hence detect the conflict. A simple probability calculation reveals that the probability of a non-empty intersection between sub-teams, i.e., the probability of an overlap between the teams, is:

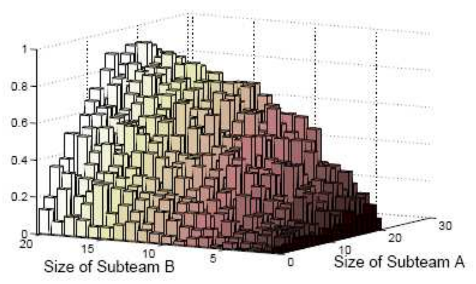
$$Pr(overlap) = 1 - \frac{{}_{(n-k)}C_m}{{}_nC_m}$$

where ${}_aC_b$ denotes a combination, n = number of agents, k = size of $sub-team_i$ and m = size of $sub-team_j$.

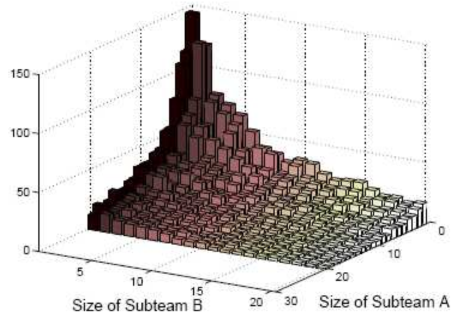
Hence, the sizes of the sub-teams are critical to the probability of overlap. For example, if $|sub-team_i| = |sub-team_j| = 20$ and $|A| = 200$, then $P(overlap) = 0.88$, despite each sub-team involving only 10% of the overall team. Since the constituents of a sub-team change over time, this is actually a lower bound on the probability that a conflict is detected.

After a conflict is detected, the plan needs to be terminated, and the same follows with the completion of goals or recipes and irrelevant or unachievable plans. We capture the domain-specific knowledge that defines these conditions with ${}^{term}p_{recipe}$. In exactly the same way as STEAM, when any $a \in sub-team_i$ detects any conditions in ${}^{term}p_{recipe}$, it is obliged to ensure that all of the other members of $sub-team_i$ also know that the plan should be terminated. In this way, the team can ensure that $plan_i \subseteq plans(t)$, i.e., no agent believes the team is performing any plan that it is not performing.

To understand the functionality of the associate network, simulations were run to see the effect of having associates on a dynamically changing sub-team. We wanted to demonstrate that, if the sub-teams have common members (associates), then conflicts between sub-teams can be detected more easily. Two sub-teams, each composed of one to 20 members, were formed from a group of 200. For each sub-team size, members were chosen at random and then checked against the other sub-teams for any common team members. Figure 17a



(a)



(b)

Figure 17: Sub-teams (a) The probability of having at least one common agent vs. sub-team size. (b) The average number of times that agents need to be replaced in order to have at least one common agent.

shows the calculated percentage of team member overlap when the sub-teams are initially formed during the simulation. This graph matches closely with the calculated $Pr(overlap) = 1 - \frac{{}_{(n-k)}C_m}{{}_nC_m}$. Since sub-teams are dynamic, in the case that both teams are mutually exclusive, a team member was chosen at random to replace a current sub-team member. Figure 17b shows the average number of times that team members needed to be replaced before a common team member was found.

8.2 PERSPECTIVE AGGREGATION

Some coordination algorithms or activities require that each member of the team builds an accurate view of the team's state. For example, the way that an individual agent uses shared resources, such as communication bandwidth or fuel, should depend on the team's overall need for such resources. On the other hand, to enable adjusted autonomy [82], an accurate view of the multi-agent team's overall properties, such as the percentage of team members that are busy or the overall level of remaining fuel of a UAV team, are required.

Technically, we can think of every member of the team as having a local value for some variable (e.g., their local need for some resource) and needing to know the average value of that variable across the whole team (e.g., the average need for some resource). Formally, each agent has some variable v. The perspective aggregation problem is for each agent to know $\bar{v} = \frac{\sum_{a \in A} v}{|A|}$.

One way of building up a perspective across the team is to have a small number of *propagators* move from agent to agent, taking the current perspective from one agent and adding it to the current perspective for the next agent. If there are multiple propagators simultaneously moving around the team, perspectives build up very quickly. Notice that it is typically infeasible for a propagator to record precisely which agents it has collected values for because it would need to record all the agent IDs as it moved from agent to agent. In a large team, this imposes an unreasonable communication load. However, because the propagator does not know precisely which agents it has visited, some will be visited repeatedly and their values counted repeatedly, distorting the average results.

A simple model of how quickly these perspectives build up can be straightforwardly created by considering how many other values each agent knows about. Before any propagators move, each agent knows only their value. Thus, the average number of values known by each agent $Avg(v, 0)$ is one. When a propagator moves, one of the agents gets to know one new value, hence $Avg(v, 1) = 1 + \frac{1}{|A|}$. In general, the average number of values known by an agent after move t of the propagators is $Avg(v, t) = Avg(v, t-1) + \frac{Avg(v,t-l)}{|A|}$. Because propagators collect information as they move, $Avg(v, t)$ rapidly grows with t. Figure 18 shows $Avg(V, t)$ for a team with 500 members. The x-axis shows the number of propagator moves divided by the number of agents and the y-axis shows $Avg(V, t)$ on a logarithmic scale. The figure suggests that perspective aggregation is not a communication-intensive task for a team, even one with relatively few edges.

The average value does not capture two key aspects of the perspective aggregation problem. First, nodes with higher degrees will be visited more often by a randomly moving propagator than nodes with lower degree. This effect can be modeled by changing the denominator in $\frac{Avg(v,t-l)}{|A|}$ to be $\rho|A|$, i.e., agents with a higher-than-average degree will have $\rho < 1.0$ and those with a lower-than-average degree will have $\rho > 1.0$. Thus, networks with many nodes with a low degree are likely to perform poorly on this task. Second, clearly many of the values an agent gets to know will be repeated. The distortion caused to the agent's

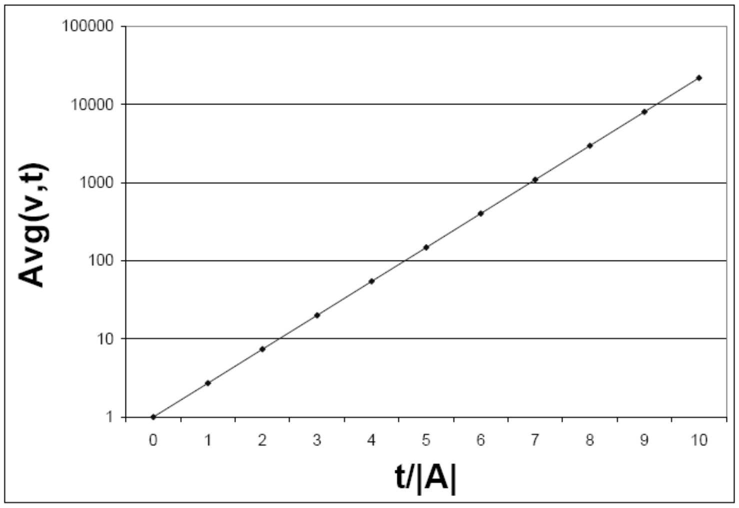


Figure 18: The average number of samples each agent has (y-axis) after a propagator has moved a fixed number of steps (x-axis). The y-axis has a logarithmic scale.

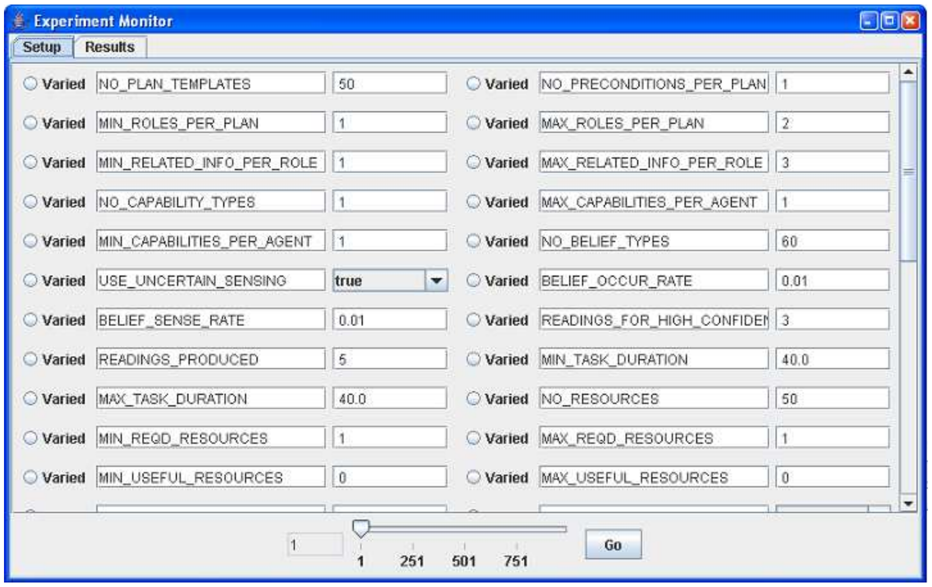
perspective by the repeats will be proportional to the relative rates at which repeats occur, i.e., if some values are repeated many times and others are not, the agent's perspective will become very distorted. An agent will likely get to know about another agent's value more often if that agent is close to it in the network than if it is far from it. Thus, networks with higher width are likely to perform poorly on this task.

9.0 EVALUATION

The evaluation of my proposed token-based algorithm incorporates evidence from a set of fidelity experiments. I have designed an abstract simulator that runs very fast and allows us to change as many parameters as possible in order to test the algorithm. The objective of my experiment is to coordinate a few hundred agents. The multi-agent team sizes in my experiment range from 50 to 1000.

There are four sections in this evaluation. First, to show the feasibility of my approach, I compare my integrated token-based coordination algorithm with the baseline of random token movement and the individual coordination algorithm, which only applies to information sharing, role assignment, and resource allocation. Because the complexity of large-scale multi-agent team coordination, I cannot evaluate my approach in all domains, but four typical application domains have been selected. Second, I compare my algorithm with the centralized market-based approach that maximizes team utility. I demonstrate the efficiency of my approach in terms of the trade-off in communication cost. Third, I investigate the robustness of my approach with respect to two questions: (1) how well the token-based approach solves plan confliction and (2) how well the algorithm performs if the definition of relevance between tokens is imprecise. Fourth, I investigate the influences of social network topologies on my algorithm.

In all of the experiments listed below, the experiment results are measured in two way: (1) the multi-agent team earns more rewards in trading of less cost of messages and (2) a message is defined as one movement of a piece of token.



○ Varied	NO_PLAN_TEMPLATES	50	○ Varied	NO_PRECONDITIONS_PER_PLAN	1
○ Varied	MIN_ROLES_PER_PLAN	1	○ Varied	MAX_ROLES_PER_PLAN	2
○ Varied	MIN_RELATED_INFO_PER_ROLE	1	○ Varied	MAX_RELATED_INFO_PER_ROLE	3
○ Varied	NO_CAPABILITY_TYPES	1	○ Varied	MAX_CAPABILITIES_PER_AGENT	1
○ Varied	MIN_CAPABILITIES_PER_AGENT	1	○ Varied	NO_BELIEF_TYPES	60
○ Varied	USE_UNCERTAIN_SENSING	true	○ Varied	BELIEF_OCCUR_RATE	0.01
○ Varied	BELIEF_SENSE_RATE	0.01	○ Varied	READINGS_FOR_HIGH_CONFIDEI	3
○ Varied	READINGS_PRODUCED	5	○ Varied	MIN_TASK_DURATION	40.0
○ Varied	MAX_TASK_DURATION	40.0	○ Varied	NO_RESOURCES	50
○ Varied	MIN_REQD_RESOURCES	1	○ Varied	MAX_REQD_RESOURCES	1
○ Varied	MIN_USEFUL_RESOURCES	0	○ Varied	MAX_USEFUL_RESOURCES	0

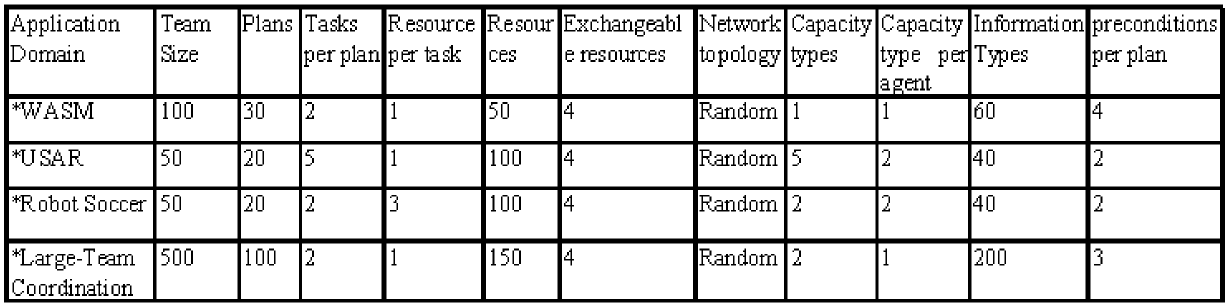
Figure 19: CoordSim allows us to test the coordination algorithm by varying many parameters

9.1 ABSTRACT SIMULATION ENVIRONMENT

I have designed an abstract simulator called CoordSim [103]. This simulator is capable of simulating the major aspects of Coordination, including sensor fusion, plan management, information sharing, task assignment, and resource allocation. CoordSim abstracts the environment by simulating only its effects on the team. Uncertain sensor readings are received randomly by some agent or agents in the team at a parameterizable rate. Agents cannot receive any domain knowledge unless they sense it themselves or are "told" by a teammate. The physical resources required for tasks are simulated and allow only one agent to access them at any given time. There is no cost for transferring resources, and resources cannot be consumed or lost. I simulate the spatial layout of tasks, which are distributed randomly in an open environment. In these experiments, all agents move at an equal speed. All agents are allowed to "think" and "act" at each time-step, although the effects of their "actions" are abstractly simulated such that they only take one time-step. Communication is implemented via object passing, making it very fast. Reward is simulated as being received by the team when a task is allocated to one of its agents. The agent's simulated location is at the task location, and the agent has exclusive access to required resources. The most prominent advantage of CoordSim is that it allows a large number of parameters to be varied and also allows statistics to be recorded, such as the number of rewards and token movements. These variations and recordings help to verify my approach. An interface of this simulator is shown in Figure 19. There are more than 20 parameters that can be varied, covering the major aspects of scalable coordination.

Because of the complexity of the scalable-team coordination problem, there are many parameters that can be varied and tested. Specially, In this thesis, I am mainly interested in the parameters that contribute most to my algorithm.

- Communication failure describes the probability that the message will be lost.
- When communication cost is high, the team will receive a much lower reward if more communications are used to reach the same team goal.
- The communication processing rate is the number of messages that an agent can pass per second. If there are more messages, they have to be sent in the next second. In my research, each token will be passed one hop with one message sent.
- Real-time control means that a task or a plan has to be carried out in a short period of time. Therefore, any task token cannot travel very far.
- Task importance describes how the reward is calculated. For example, the goal of a UAV team is to destroy as many enemy vehicles as possible, and missing one is not a major concern. For USAR, however, missing one victim means that the team has failed, so every task and plan must be allocated.
- The team size is the number of agents on the team.
- A heterogeneous team describes the number of capability types among team members. In USAR, heterogeneous robots always have many different capabilities.
- Team robustness is defined by the failures of team members–for example, if one WASM is killed or if one robot in USAR stops working.
- The Exclusive Resource Requirement describes the number of required resource types.
- Plan complexity defines the number of pre-conditions to activate each plan.



Application Domain	Team Size	Plans	Tasks per plan	Resource per task	Resources	Exchangeable resources	Network topology	Capacity types	Capacity type per agent	Information Types	preconditions per plan
*WASM	100	30	2	1	50	4	Random	1	1	60	4
*USAR	50	20	5	1	100	4	Random	5	2	40	2
*Robot Soccer	50	20	2	3	100	4	Random	2	2	40	2
*Large-Team Coordination	500	100	2	1	150	4	Random	2	1	200	3

Figure 20: Four typical application domains with different features.

9.2 FEASIBILITY OF THE TOKEN-BASED APPROACH

In this section, I will show the experiment results to verify the feasibility of the token-based approach. Four configurations of the algorithm were compared. In the first configuration, agents passed tokens randomly if they did not keep them. In the next second and third configurations, local reasoning model updating is applied to only some types of tokens (e.g., information, resource, or role) with no updating of the other types. The fourth configuration provided integrated coordination using tokens of each type to update agents' local models for routing tokens. My hypothesis for these experiments is that the integrated algorithm will outperform the random and partial coordination algorithm with any size of multi-agent teams. Moreover, the single algorithm will perform well at different stages. Information sharing will take effect at the beginning, while role and resource allocation perform in the latter stages.

Because of the complexity of large-scale multi-agent team coordination, I cannot evaluate my approach in all relevant domains, but I have selected four typical application domains. In addition, I have abstracted and varied parameters for these four different domains according to their typically different features and applications: urban search and rescue (USAR) [17], controlling WASMs, RoboCup [66], and strategy game/decision support simulation/scheduling [4]. For example, in USAR, coordination is mainly focused on heterogeneous teams and each team-oriented plan template has only one or two roles that need to be assigned. I summarize these domains according to the features listed below:

- UAV: Middle team size, middle number of plans, middle number of tasks per plan, low number of exclusive resources, low number of capability types, and high number of plan pre-conditions. In this domain, information sharing is the most difficult task.
- USAR: Small team size, middle number of plans, high number of tasks per plan, middle number of exclusive resources, and high number of capability types. In this domain, task allocation is the most difficult task.
- RoboCup: Small team size, middle number of plans, middle number of tasks per plan, low number of exclusive resources, and low number of capability types. In this domain, resource allocation is the most difficult task.
- Large-team coordination, such as video games or off-line scheduling: large team size, middle number of plans, middle number of tasks per plan, middle number of exclusive resources, and middle number of capability types.

Table 20 summarizes the specific settings in CoordSim for these different domains. For example, in the first domain of coordinating UAVs, I simulated a group of 100 distributed UAVs searching a hostile area. The network topology was that of a random network where each UAV had, on average, four associates. Simulating automatic detection rates, 60 pieces of information were randomly sensed by UAVs and passed around the team. Thirty plan instances, each with four independent pre-conditions, were given to the team. After a plan was initiated, tokens for the two roles needed to realize the plan were circulated through the associate network. To accept a role, an agent must be close to the region that the role requires and must have access to resource tokens for airspace at the role allocation location. There were 30 abstracted resources around the team, and each of these was interchangeable with four other resources. Each UAV needed to obtain one required resource related to its task before the task could be performed. If a role was successfully executed, which could take one to ten time-steps, a reward was credited to the team. The reward amount for completing a role depended on the importance of its plan and the cost for that agent to travel from its current location to the location where the role needed to be executed (e.g., a UAV must fly

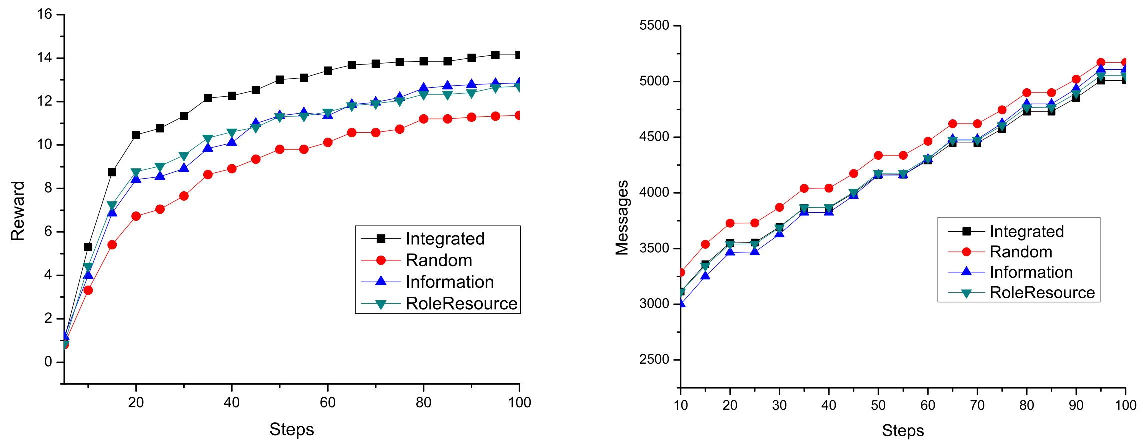


Figure 21: Experiment results for the domain of *UAV coordination: reward gain and message cost over different time-steps.

to the location of the enemy tank to destroy it). For each domain, other parameters varied slightly–for example, depending on the team size, the information sensing rate for each agent varied from 0.02 to 0.006 and the allowed TTL for each information token from eight to 20.

Please note, in my experiment, because of the nature of the abstracted simulator, it only involves configurations that reflect characteristics of the listed application domains and that tests were not conducted in those domains themselves. Additionally when the domains are referred to, a * will accompany the name to signify similarity rather than identity. For example, I will refer the UAV test domain as *UAV to identify that my experiment will perform in an abstracted domain similar with coordinating UAVs.

in this section, for each application domain, two experiments were independently performed. In the first experiment, all of the information is sensed at the initial steps and the experiment runs for a small number of steps. This experiment verifies my hypothesis that information tokens are critical during the early stage of plan instantiation, and knowing who is initiating plans will gather more rewards during these initial steps. But resource and role tokens are more critical after a plan has been initiated and contribute to more rewards at later stage. In this experiment, I investigate the number of rewards and the number of

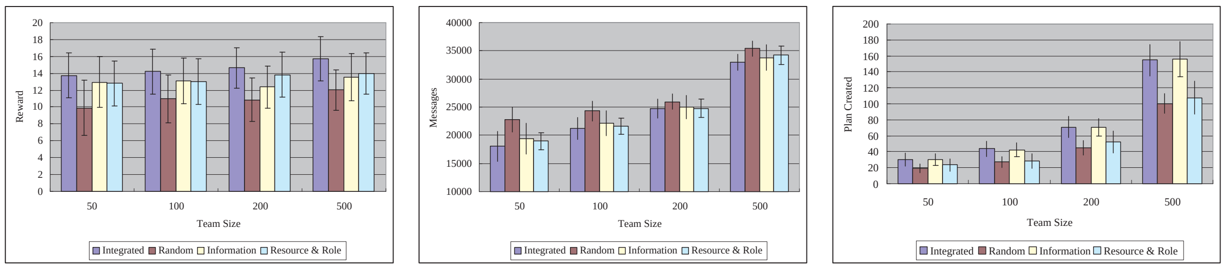


Figure 22: Experiment results for the domain of *UAV coordination: reward gain, message cost, and plan incitation over different sizes of teams (50, 100, 200, and 500).

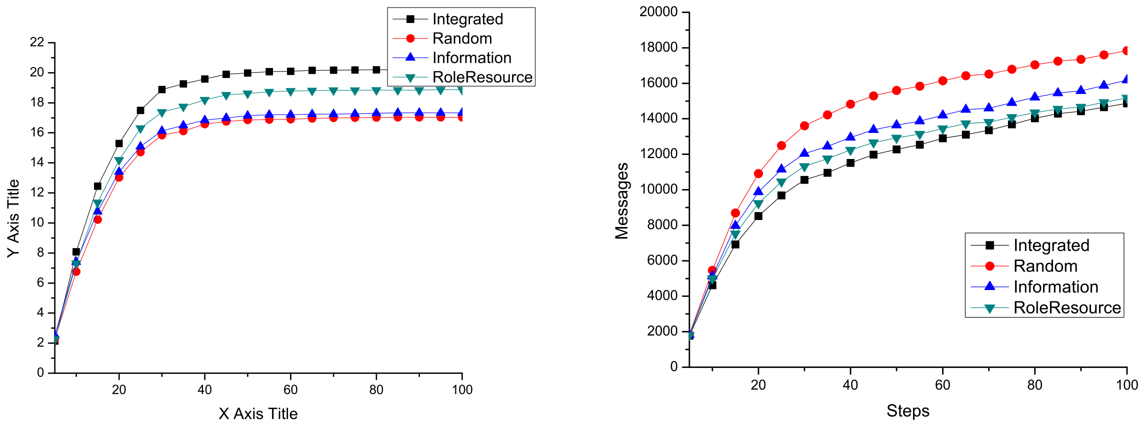


Figure 23: Experiment results for the domain of *USAR: reward gain and message cost over different time-steps.

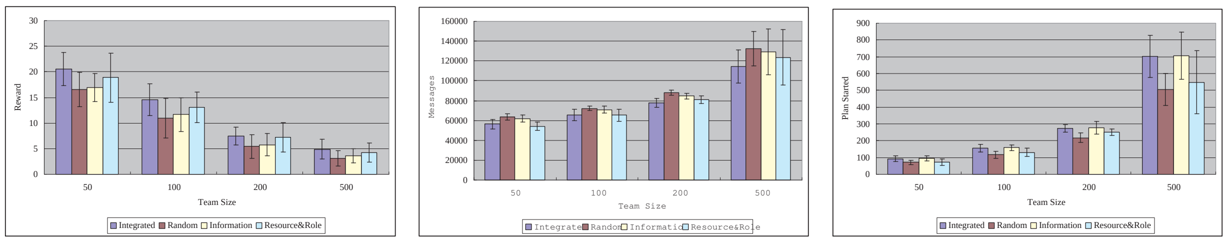


Figure 24: Experiment results for the domain of *USAR: reward gain, message cost, and plan initiation over different sizes of teams (50, 100, 200, and 500).

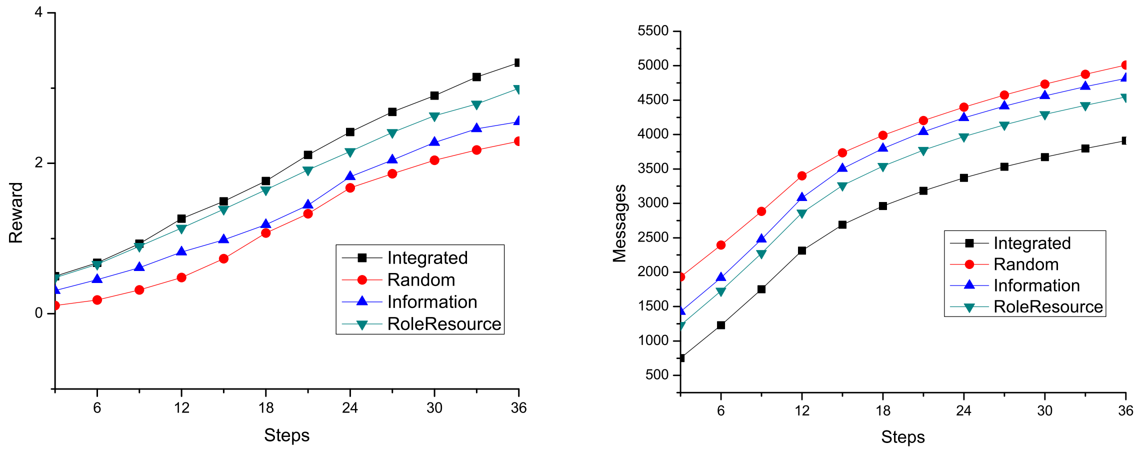


Figure 25: Experiment results for the domain of *RoboCup: reward gain and message cost over different time-steps.

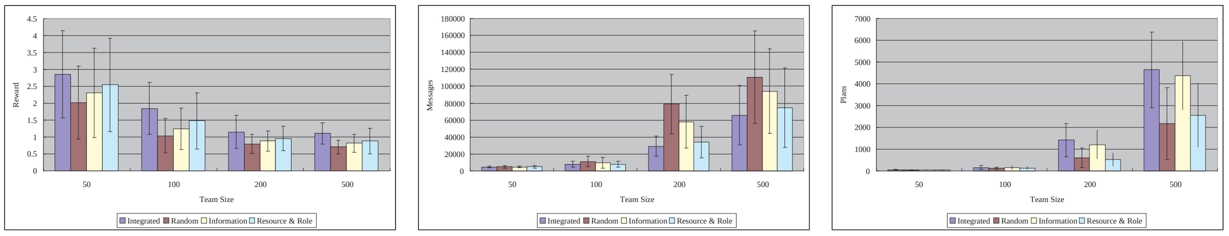


Figure 26: Experiment results for the domain of *RoboCup: reward gain, message cost, and plan initiation over different sizes of teams (50, 100, 200, and 500).

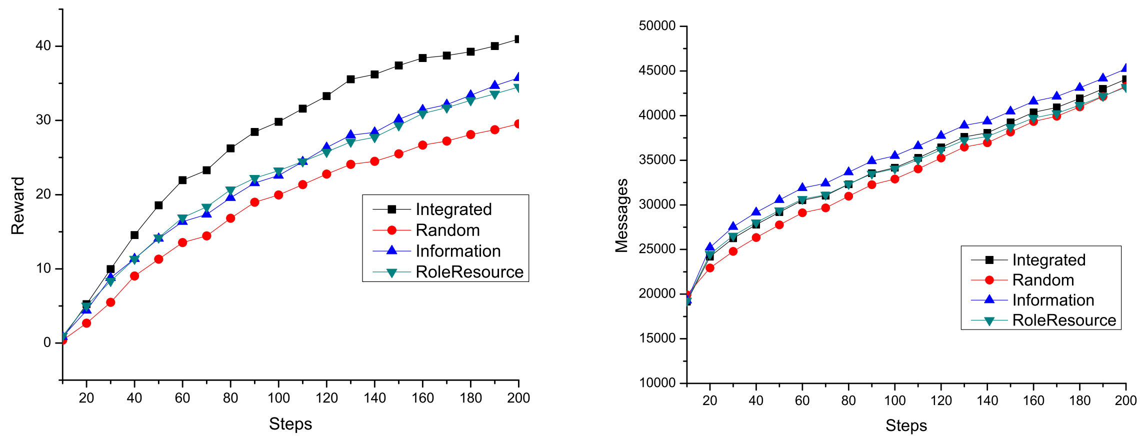


Figure 27: Experiment results for the domain of *large-team coordination: reward gain and message cost over different time-steps.

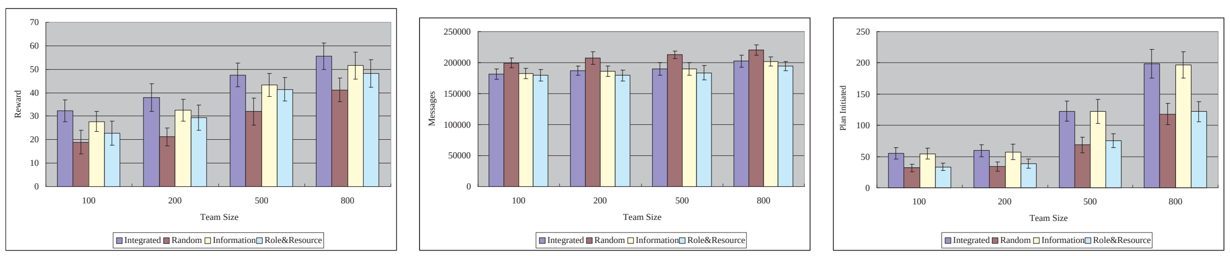


Figure 28: Experiment results for the domain of *large-team coordination: reward gain, message cost, and plan initiation over different sizes of teams (100, 200, 500, and 800).

messages (one message per token move) after a specific number of time-steps.

In the second experiment, the settings are more realistic. Information is sensed randomly, and experiments last longer. The objective of this experiment is to find whether the integrated token-based algorithm can earn more rewards with fewer messages. More plans will be initiated. This experiment runs different sizes of multi-agent teams in four different domains. All of the experiment results shown below are based on 100 runs.

From Figure 21 to 28 shows the experiment results over the four different application domains. Clearly, the information-sharing algorithm outperforms the role and resource allocation algorithm in figure 21 and 27. This is not so prominent in 23 and 25 because information sharing is not difficult in these two domains. In contrast, resource and role allocation works better in the later stages. Both of them are uniformly better than random with more rewards and a lower cost of messages at any stage, for any team size, and in any domain. This also demonstrates that my intelligent algorithm works on each single algorithm. Across these eight graphs, the integrated token routing algorithm outperforms all of the other algorithms at any stage, for any team size, and in any application domain by earning more rewards, incurring lower costs for messages, and initiating more plans.

9.3 COMPARING WITH MARKET-BASED COORDINATION ALGORITHMS

Although in the last section, I have shown that my algorithm is feasible for efficient coordination within different sizes of team, the difference between this algorithm and the optimal solution is still unknown. In this section, I make a systematic and scientific comparison with the coordination algorithm, which approaches optimal performance. In my thesis, I will compare a market-based approach because it is the most popular centralized approach. By designing the central auctioneer, this approach is capable of finding a policy that nears optimal performance.

9.3.1 Algorithm Design

I will use CoordSim to investigate the relative strengths of three distinct coordination approaches: auction-based coordination [26], my token-based coordination, and a hybrid of the two [103]. The hybrid algorithm is developed not to be superior to the other two, but to investigate a hypothesis about the observed relative strengths of the first two algorithms. The comparison has two additional objectives: (1) to determine the strengths of each approach and (2) to find out whether their strengths can be combined into a hybrid approach.

Based on an analysis of previous literature [104, 5], several hypotheses can be formulated regarding the relative performance of these algorithms. Auctions are focused on maximizing overall utility by taking into account the *bids* of all team members [5]. Token-based algorithms are focused on scalability; they minimize communication, sometimes at the expense of overall utility. Thus, the clearest hypothesis that emerges is that auctions will involve more communications than token-based algorithms, but will result in better allocation of tasks and resources. More subtly, the performance advantage of an auction should be most pronounced when small changes in allocations lead to large differences in performance, as in typically highly constrained cases. The token-based algorithms should maximize their communication advantage when the probabilistic models that they rely on are most advantageous, as in weakly constrained cases.

Unfortunately, the overlap in coordination tasks that can be performed by both tokens and auctions is limited to role and resource allocations, hence the focus of the comparison on these capabilities. In my experiments, other tasks required for coordination, such as initiating joint tasks and sharing information, will always be performed by the token-based algorithm.

9.3.1.1 Market-Based Algorithm My implementation of the market-based approach will be based on TraderBots [26] with adaptations when necessary in order to make a comparison possible. In this approach, one agent acts as the auctioneer, and both tasks and resources are treated as merchandise. Agents bid for either single items or combinatorial sets of items in order to maximize their own utility. The auctioneer maximizes its utility by "selling" this "merchandise." In this approach, Sandholm's winner determination algorithm [78] is used to determine the auctioneer's allocation of tasks and resources. Because of the centralized position of the auctioneer, it develops a complete knowledge of how agents will use a task or resource if allocated. Thus, the auctioneer can perform assignments to maximize team utility. Note that several constraints also apply to this approach. To be fair to all the bidders, the auction must last for a fixed period of time. Agents are allowed to bid for resources until tasks have been allocated to them. Moreover, to prevent deadlock in resource allocation, agents are only allowed to bid for resources for their *first* pending task.

9.3.1.2 Hybrid Algorithm The hybrid algorithm works in the following way. The auctioneer algorithm runs exactly as before, except that, instead of broadcasting announcements of auctions, "auction tokens" are created. All agents have a probabilistic model of the team state, just as all agents do in the token-based approach. The auction token is then intelligently routed, via the token-based algorithms, only to the agents that are most likely to submit the best bids. If an agent can submit a better bid than the lower bound estimated by the auctioneer (and represented on the token), it does so; otherwise, it passes the token on. The token stops moving after it has visited only a small number of team members but, if the intelligent routing works well, it will have visited only the agents able to submit the best bids. The auctioneer determines the winner of the auction and allocates tasks and resources as described in the basic auction case. My hypothesis is that the hybrid approach should reduce communication over the basic auction by targeting only the potential best bidder, reducing computations by limiting the number of bids that the auctioneer must deal with, and improving allocations over the token-based algorithm by allowing centralized allocations to be performed by the auctioneer. However, because of the centralized auctioneer, it will still use more messages than the token-based approach. Furthermore, because it does not solicit bids from all agents, it will not find allocations as well as the auction approach.

9.3.2 Experiment Settings

The basic experiment settings in CoordSim are configured as follows. There are 100 agents to perform 50 tasks with 50 resources. Each task requires only one resource, which is interchangeable with four other resources. In the default set-up, there is only one type of capability required and all agents have a non-zero value for this capability (i.e., all agents are at least somewhat capable of all tasks). Auctions are held open for 50 time-steps, and the task and resource tokens are allowed to move until accepted. The information sensing rate for each agent is 0.01. The initial threshold on a task token is 0.5, meaning that the task will not be accepted by an agent until its capability to perform the role is greater than 0.5. We measured two key statistics required to support or refute our hypothesis about the algorithms. "Reward" is the sum of rewards received by each agent. "Messages" is the number of times that agents communicated, either between themselves or with the auctioneer. The "messages" count indicates messages sent to perform sensor fusion, plan initiation, and information sharing. Simulation runs for 1000 time-steps. The experiment results below are based on 100 runs.

9.3.3 Heterogeneous Teams

In the first experiment, we examined team performance by varying team composition and the capabilities required to perform tasks. For example, in an emergency response experiment, some agents might only be able to fight fires while others might only be able to provide medical treatment. As capabilities grew more varied, fewer agents are available to perform particular tasks. In this experiment, we varied the number of capabilities from three to 30. In the most heterogeneous condition, only three agents on average are capable to performing a task.

The experimental results in Figure 29 show that, for heterogeneous teams, auction and hybrid approaches earn fewer rewards as the team becomes more heterogeneous because there are fewer agents able to compete for the more specialized tasks. The advantages of the auctioneer's team-wide maximization of utility decrease as there are progressively fewer feasible alternative bids. In contrast, rewards for the token-based approach remain almost

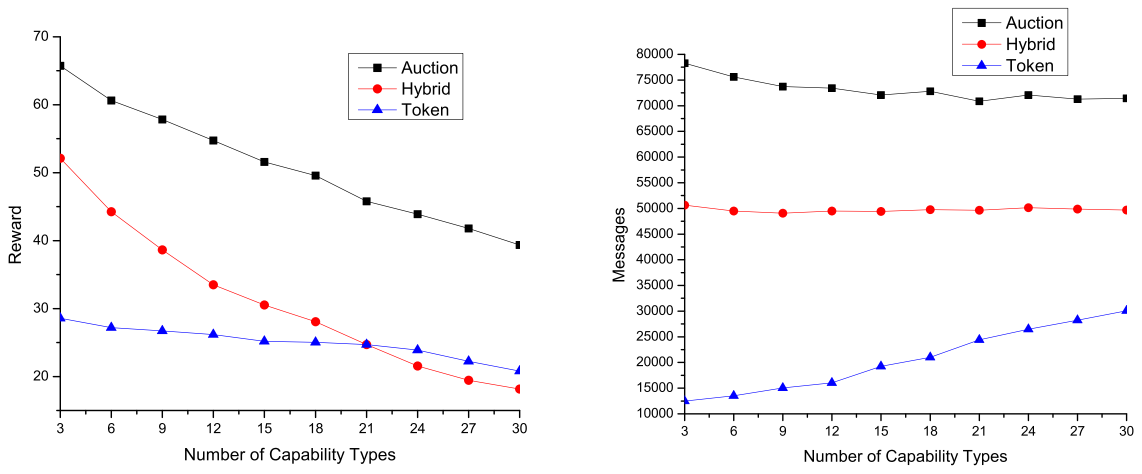


Figure 29: Coordination algorithm comparison: heterogeneous teams

flat with increasing specialization. We propose two reasons. One is that the token-based approach greedily finds a reasonable solution rather than searching for the optimal solution. The other reason is that, by passing a higher number of tokens around the network and making use of the relevance between them, the intelligent routing algorithm gains a better knowledge of how to route tokens. This results in an increasing number of messages but an equal number of rewards.

9.3.4 Intensive Tasks Allocation

In the second experiment, we investigated team performance when many tasks needed to be performed. To increase their number of rewards, teams were required to perform tasks and allocate resources as rapidly as possible. In this study, we varied the number of tasks of each plan to be finished between one and ten. All of the available roles to be accomplished by the team ranged from 50 to 500. After 1000 time-steps, the accumulated reward and message count were recorded, as shown in Figure 30.

All three approaches performed more tasks in order to receive a higher number of rewards. As expected, the auction approach attained a higher number of rewards than the

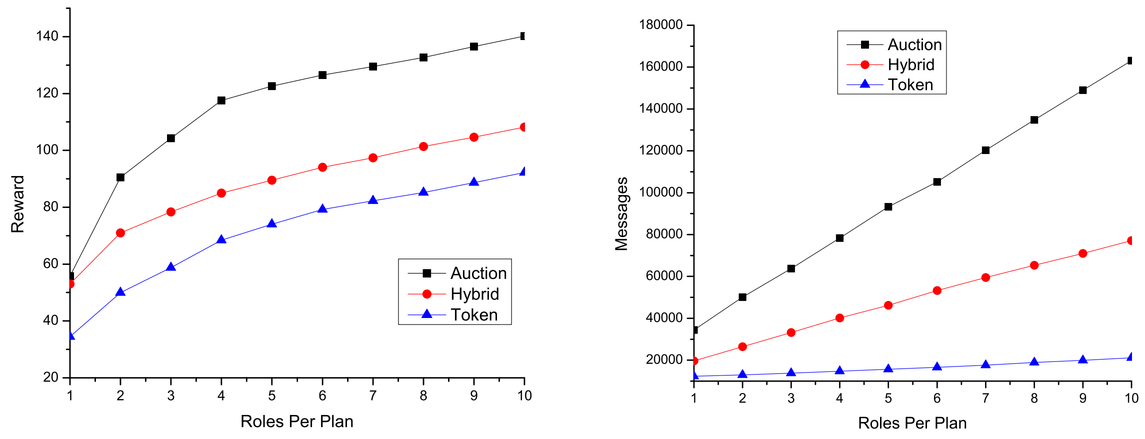


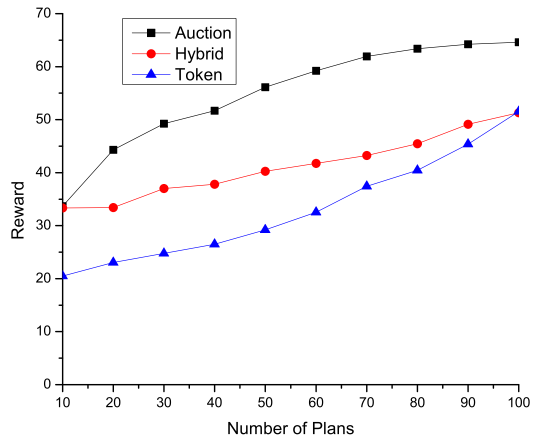
Figure 30: Coordination algorithm comparison: intensive tasks allocation

hybrid and token-based approaches. Considering both the number of rewards and messages, however, the token-based approach performs well by matching more than half of the number of rewards obtained by the auction at only one-tenth of the communication cost. Moreover, the hybrid approach makes a good trade-off between earning rewards and keeping the communication cost low as well. The reason that these approaches performed so well with so little communication overhead is that the intelligent routing algorithm limits communication to a small number of agents while ensuring that highly capable agents are always informed.

9.3.5 Time-Critical Tasks

In this experiment, similarly to the experiment described in the previous section, I limited the length of the simulation to 200 but increased all of the available roles to be accomplished by the team to range from 50 to 500.

Graph 31 shows the experiment results. All three approaches performed more tasks in order to receive a higher number of rewards. As expected, the auction approach attained a higher number of rewards than the hybrid and token-based approaches. Considering both rewards and messages, however, the token-based approach performs best by almost matching



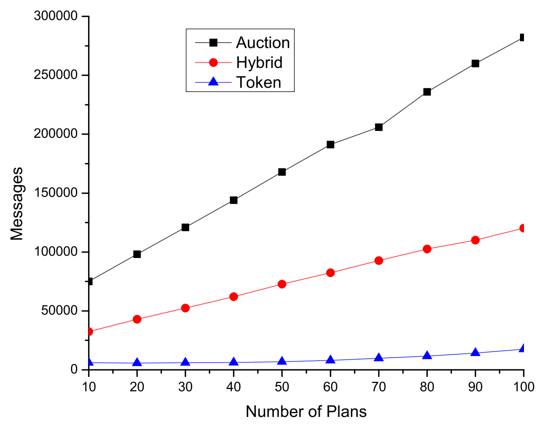


Figure 31: Coordination algorithm comparison: time-critical tasks

the number of rewards obtained by the hybrid at only a fraction of the communication cost. The reason is that the auction and hybrid approaches have the constraint of waiting to bid for the role and resource for a fixed interval. The token-based approach does not have this constraint. It can allocate the role to a capable agent very quickly.

9.3.6 Competitive Resources

The fourth experiment used 50 tasks, each of which required an average of four resources with no possibility of interchanging. As available resources increase from five to 50, competition for them declines and they become less likely to cause a bottleneck.

We hypothesized that, because resource contention in this experiment was high, the centralized control of the auction and the hybrid approach would often force agents to either bid for all four resources together or miss the task while the distributed token-based approach weakened this constraint. Experiment results are shown in 32. With the increase in the number of available rewards, three approaches receive more rewards while the auction approach receives the greatest amount of rewards, the token-based approach costs the fewest in number of messages, and the hybrid approach works best by earning a comparable number

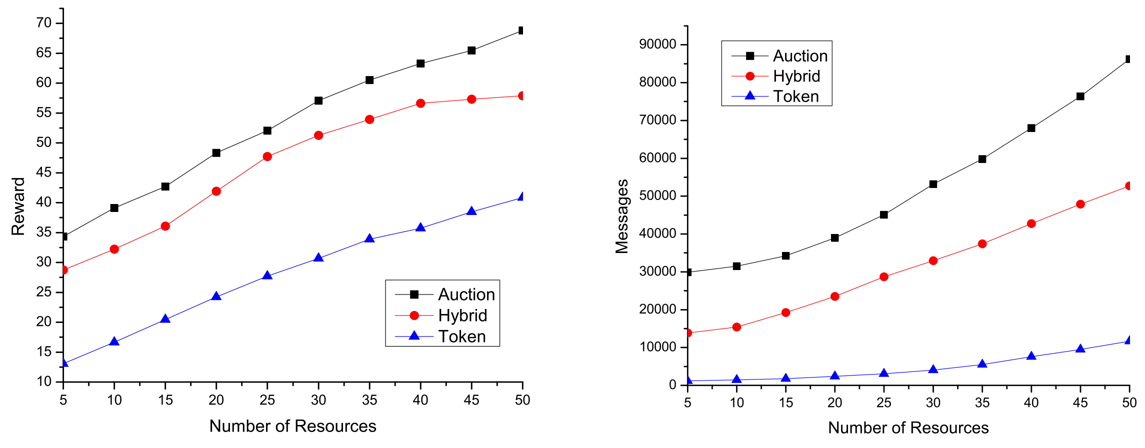


Figure 32: Coordination algorithm comparison: competitive resources

of rewards as the auction approach but incurring only half the cost of its messages.

9.3.7 Interchangeable Resources

In the fourth experiment, there were 50 tasks that each required one resource. The number of interchangeable resources was varied between two and ten.

Results are shown in Figure 33. Interchangeable resources did help the token-based approach more because agents are easier to move to the required resources when, most of the time, they are idly moving around the network. In contrast, the improvement for the auction-based approach and the hybrid approach is not as noticeable because agents still need to bid and wait for the winning notice before receiving the required resources.

9.3.8 Auction Length

In the last experiment of this section, the length of time that an auction was required to be open was varied between ten and 100 time-steps. When the auction lasts longer, all agents have to wait longer to receive a role or a required resource. On the other hand, the hybrid approach provides more opportunities to pass auction tokens to inform the right agent to

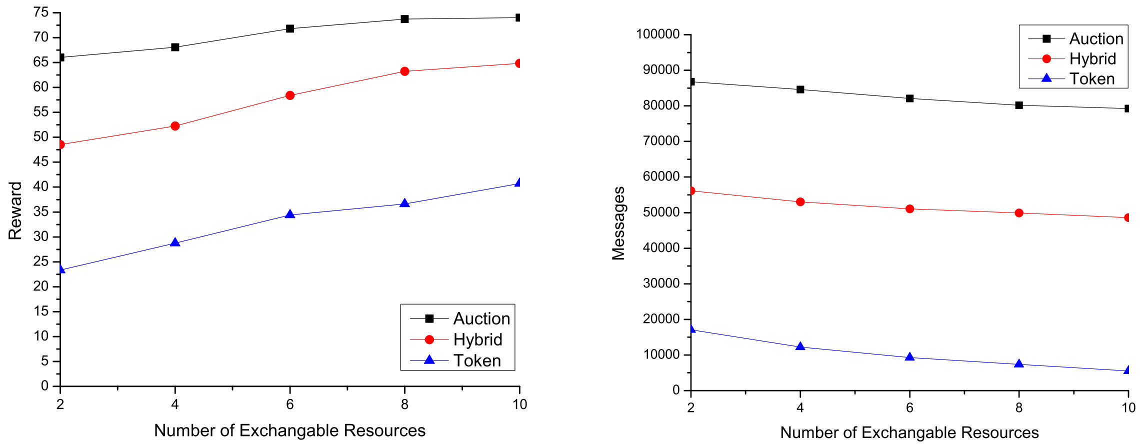


Figure 33: Coordination algorithm comparison: interchangeable resources

submit bids and to increase team performance.

Figure 34 shows that, for this experiment, the auction approach obtained a higher number of rewards for all auction lengths. These rewards, however, came at the cost of a large number of messages for short auctions. The hybrid approach, by contrast, had a much lower volume of messages and approached a comparable number of rewards with auction-based approach very quickly when the auction lasted a long time. This shows us that the intelligent routing algorithm works on auction tokens, as explained in Section 9.3.3.

9.4 IMPRECISE MODEL OF DECISION

In my approach, I make use of relevant domain knowledge to increase the efficiency of coordination. The efficiency of team coordination depends highly on the definition of relevance; thus, in this section, I investigate how robust the system is if the relevance is imprecise and waivering between intervals.

I set up an experiment with the same settings described in Section 9.2. Unlike the second experiment described in Section 9.2, however, which has a fixed relevance definition,

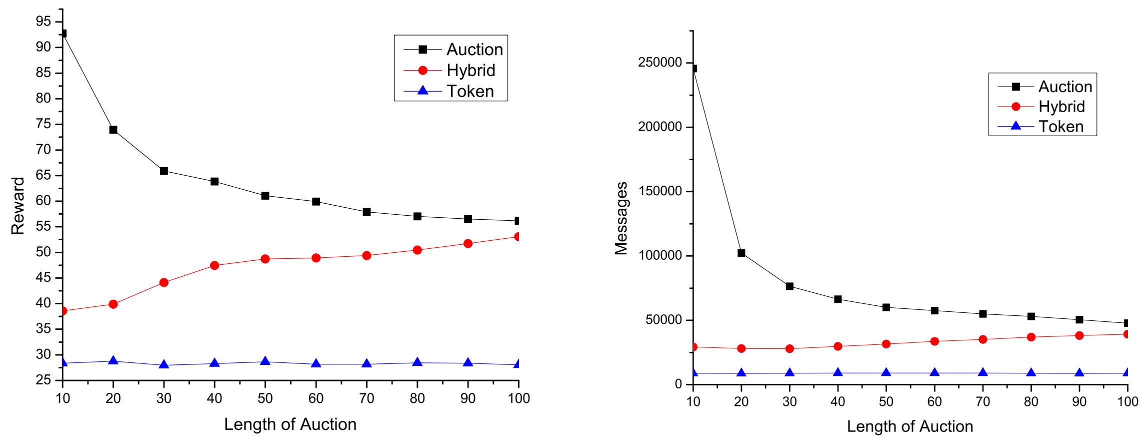


Figure 34: Coordination algorithm comparison: auction length

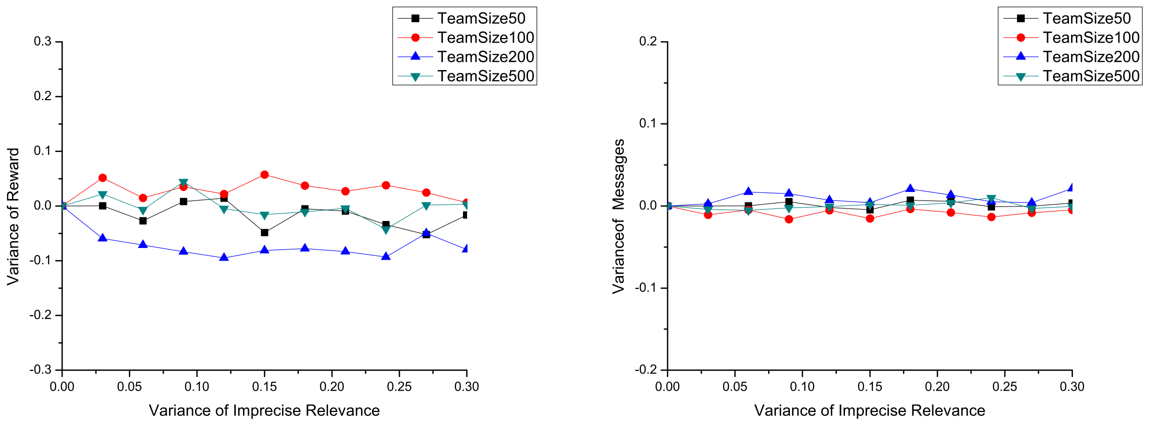


Figure 35: Effects of imprecise definition on application domain *UAV.

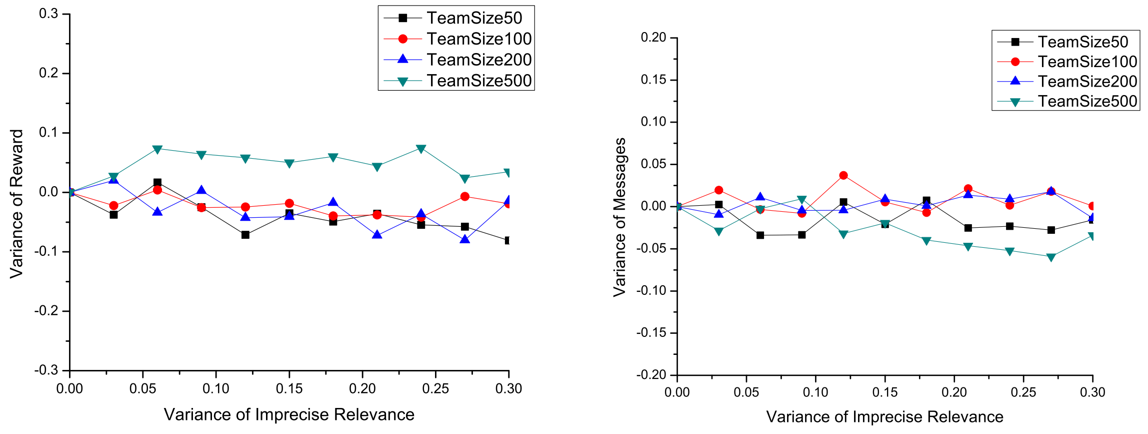


Figure 36: Effects of imprecise definition on application domain *USAR.

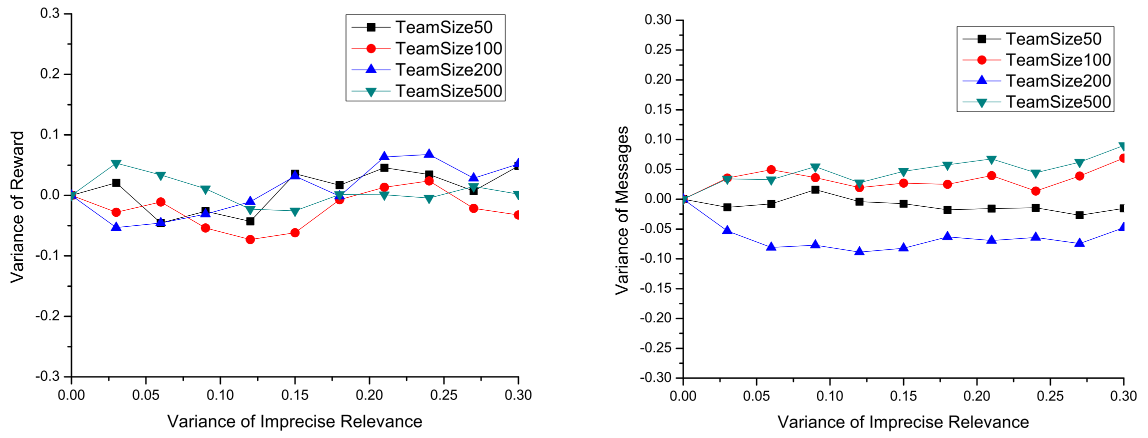


Figure 37: Effects of imprecise definition on application domain *RoboCup.

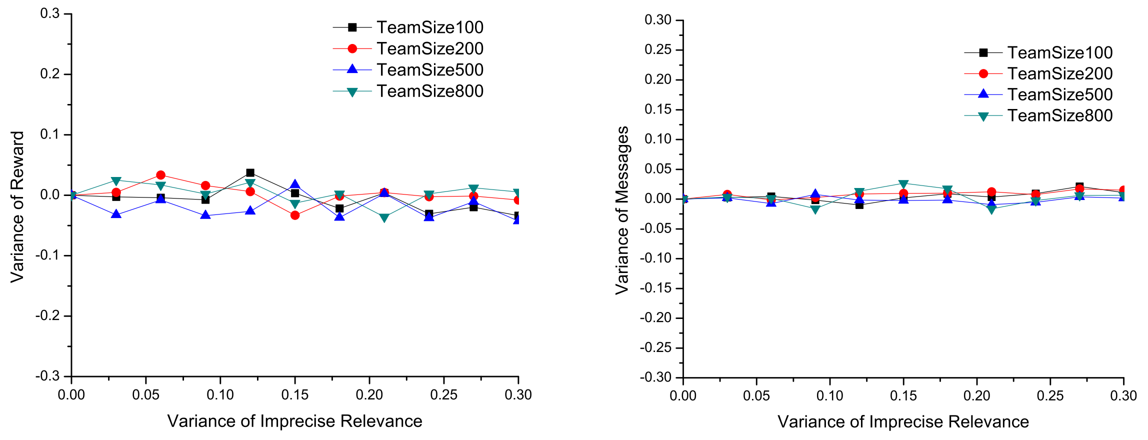


Figure 38: Effects of imprecise definition on application domain *Large-Team Coordination .

the definition of relevance in this experiment is randomly varied between a fixed interval $[-\alpha, +\alpha]$ (e.g., if $\alpha = 0.5$, the relevance will be varied between $[-0.5, 0.5]$). For each domain, the interval was varied between 0 to 30%. When the interval is zero, the relevance is fixed and relevance is no longer imprecise.

The experiment results of the effects of an imprecise definition of relevance on each application domain are presented in figure 35 through 38. In all of the experiment results, the variance of the number of rewards and the number of messages over different domains is much lower than the interval of the imprecise definition of relevance. Therefore, we can demonstrate that the integrated token-based algorithm is robust when there is an imprecise definition of relevance. Moreover, compared to the other three domains, the third domain contains the largest variance. I hypothesize that resource allocation is more dependent on a precise definition of relevance.

9.5 PLAN DE-CONFLICTION

From the experiment results given in Section 9.2, we can see that the integrated algorithm creates more plan instantiations than the actual plan templates. Therefore, many conflicted plans are created. Although I have encoded the mechanism of plan de-confliction and a conflicted plan does not count toward any reward in Section 9.2, in this section, I present my experiment results for plan de-confliction.

Table 1: Results of Plan De-conflictions

Domain	Team Size	#Plan Initiated	#Duplicated	#Detected	%Solved
*UAV	100	43.62	13.62	13.62	100%
*UAV	200	71.06	46.87	46.35	98.8%
*UAV	500	155.9	132.02	130.27	98.7%
*USAR	100	158.28	140.93	140.87	99.9%
*USAR	200	274.2	257.13	254.93	99.1%
*USAR	500	702.18	687.02	675.8	98.4%
*RoboCup	100	149.27	130.86	130.53	99.7%
*RoboCup	200	1414.57	1396.1	1391.7	99.7%
*RoboCup	500	4564.59	4548.58	4470.96	98.3%
*LargeTeam	100	55.14	16.96	16.28	96.0%
*LargeTeam	200	59.4	16.72	15.83	94.7%
*LargeTeam	500	122.68	41.29	37.86	91.7%

Table 1 shows all of the plan de-confliction results over four different application domains and different team sizes (100, 200, and 500). Because detecting conflict is too easy when a team is composed of 50 agents, these results have been excluded from this experiment. Of the remaining results, when team size is less than 200, nearly 99% of plan conflictions is detected and solved. Even in the largest team composed of 500 agents, more than 90% of plan confliction is solved. Moreover, if information sharing is not difficult in the domain, such as the third and fourth domains, the plan de-confliction rate is close to 100%. Therefore, our solution of plan de-confliction is efficient.

9.6 EFFECTS OF ASSOCIATE NETWORK TOPOLOGIES

As my initial experiment in Section 5.4 has shown, social network topology may exert an important influence on the efficiency of my token-based algorithm. In this section, the primary purpose is to determine how social network topology influences my algorithm. Similarly to the experiment described in Section 9.2, experiments in this section are based on the same four domains (i.e., *UAV, *RoboCup, *USAR, and *large-scale coordination). Four social network topologies will be tested: random network, small world network, grid-based network, and scale-free network. I present two sets of experiments. First, I expand the team size in each domain from 50 to 500 and determine how social network topology influences team sizes that range from small to large. Second, I determine whether more associates will be helpful to my algorithm. In this experiment, all of the teams consist of 100 agents and the average number of associates for each agent varies from two to ten. To make the influence of social network effects more easily observable, the probability of information sensing will be lowered (1% to 2%) and the simulation will run only 50 steps in the first three domain and 100 steps in the fourth domain. All of the other settings remain the same as described in Section 9.2.

The first experiment results are shown as 39, 41, 43, and 45. In most domains, because they possess the property of the small world effect, the random network and the basic small world network outperform the grid-based network. In some domains, because they possess the property of the scale-free effect, the scale-free network may outperform the other three social topologies. The reason is that a few agents can act as hub nodes and obtain more knowledge of how to pass a given token in the right direction. On the other hand, in some situations such as that presented in figure 43, the scale-free network produces more messages and receives fewer rewards. Here, the reason is that the hub nodes maintain many associates but the average tokens from each associate can be decreased. Therefore, the hub nodes cannot build a better model based on the previous tokens that they have received.

The results of the second experiment are shown in 40, 42, 44, and 46. On average, all of the network topologies increase their performance when agents keep an average of four associates instead of the average of two. But when the average number of associates increases further across these domains with the exception of the grid-based network, the other social network topologies' performances can become worse. An extreme example of this is a social network topology that is a complete network. I cannot represent this result in the figures because, compared with the grid-based network, it will take more than 100 times of messages to receive only a very small number of rewards. Moreover, in the fourth domain, each run takes more than two hours. These results reinforced my hypothesis that, when the number of associates per agent increase, the average number of tokens from each associate can be decreased and the agents may not maintain better decision models based on previously received tokens. This is most prominent in the hub nodes in a scale-free network. This is the reason that, in figure 42, the scale-free network's performance decreased so quickly. Conversely, the grid-based network enhances its performance in two ways. First, the average number of associates is the exact number of associates for each agent; therefore, no agents can have a large number of associates. Second, with the increasing number of associates, the average distance in the grid-based network is drastically decreased. In this way, it is much easier to pass tokens quickly to their destinations. Therefore, we have verified the importance of the small world effect on the efficiency of the present token-based approach.

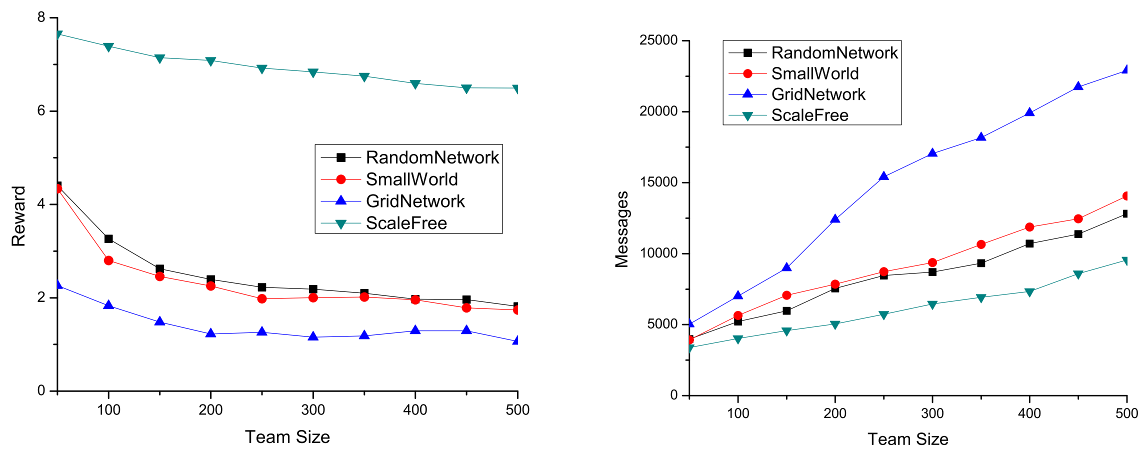


Figure 39: Application domain *UAV: social effects on different team sizes.

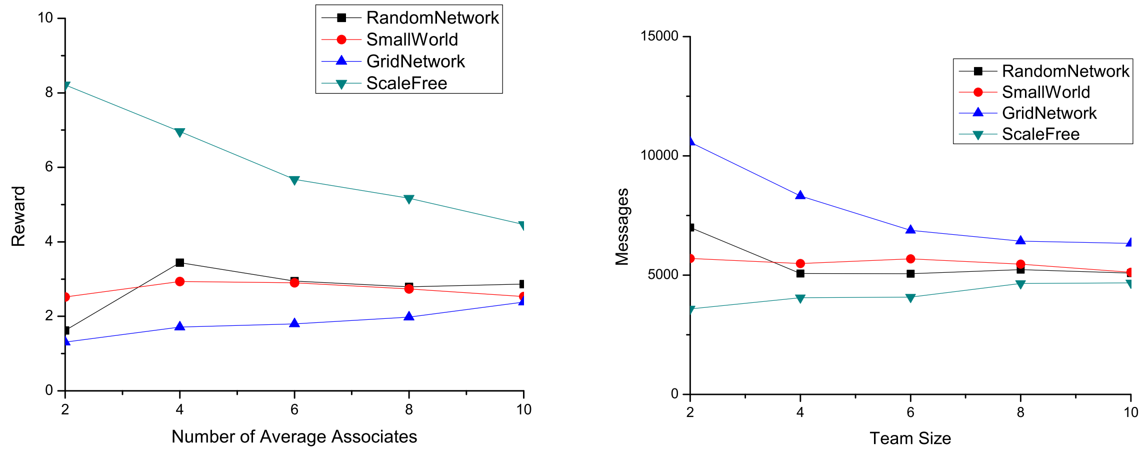


Figure 40: Application domain *UAV: social effects on different number of average associates.

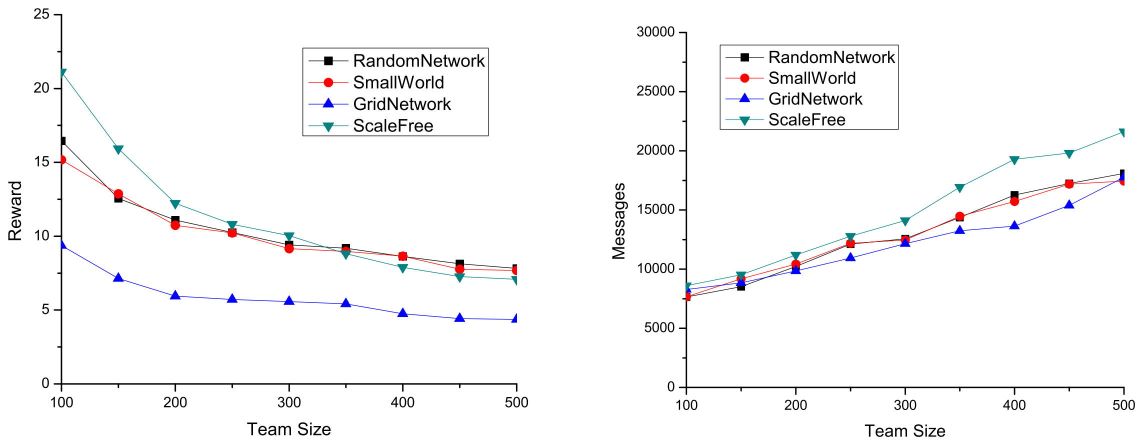


Figure 41: Application domain *USAR: social effects on different team sizes.

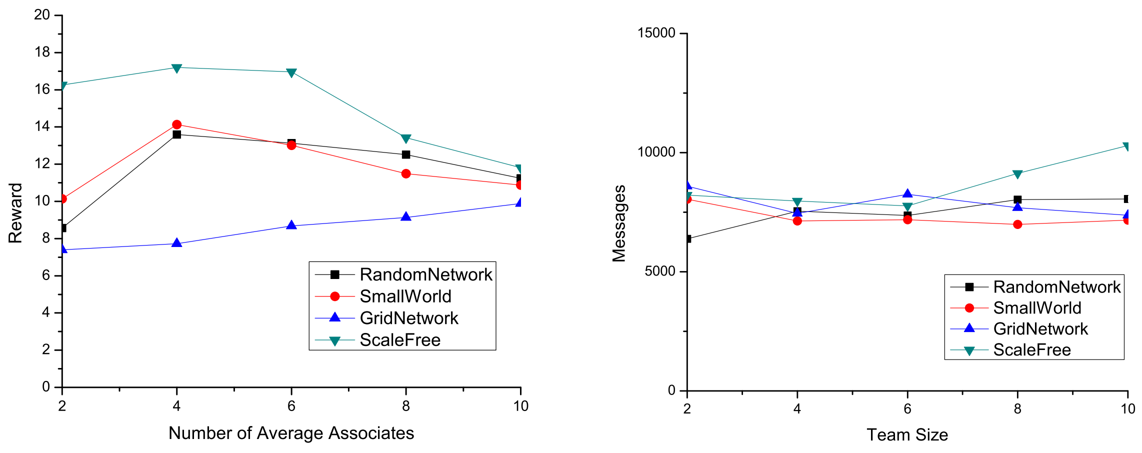


Figure 42: Application domain *USAR: social effects on different number of average associates.

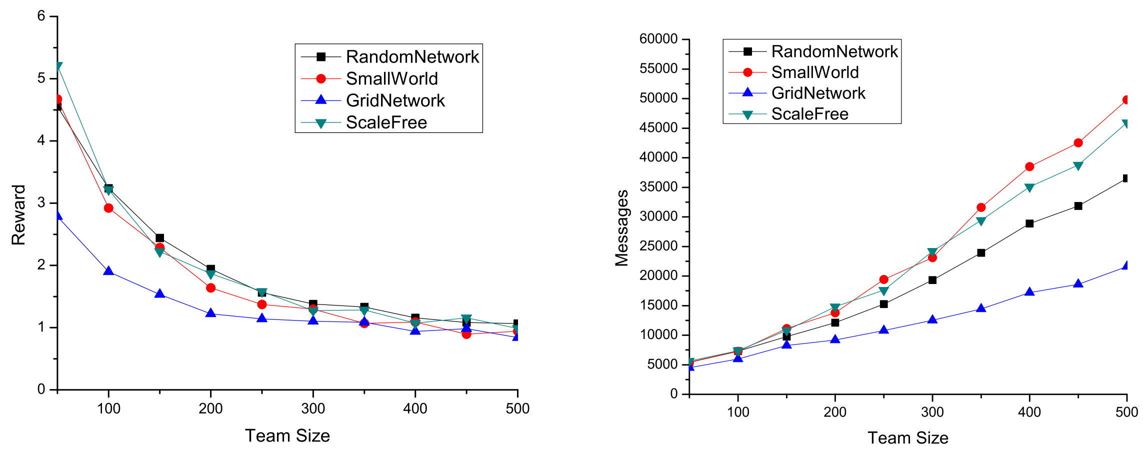


Figure 43: Application domain *RoboCup: social effects on different team sizes.

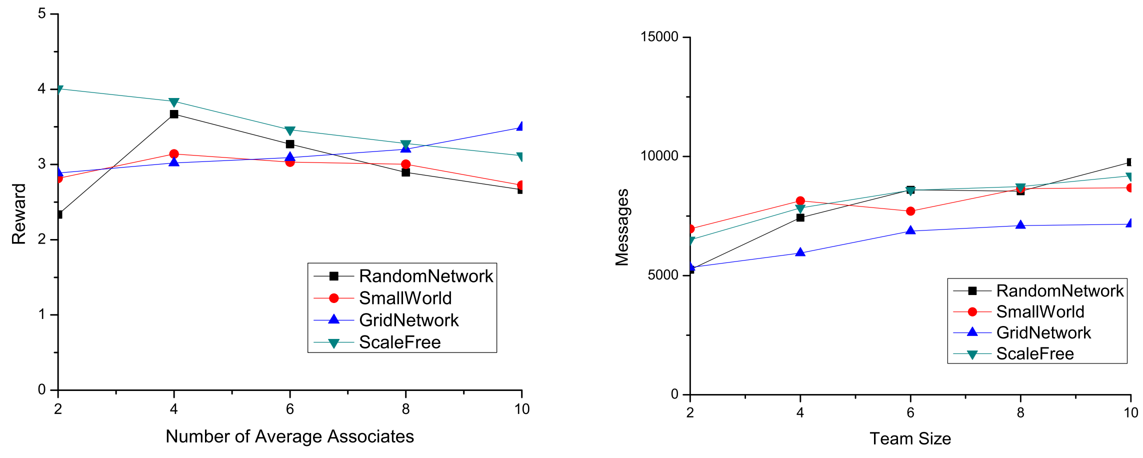


Figure 44: Application domain *RoboCup: social effects on different number of average associates.

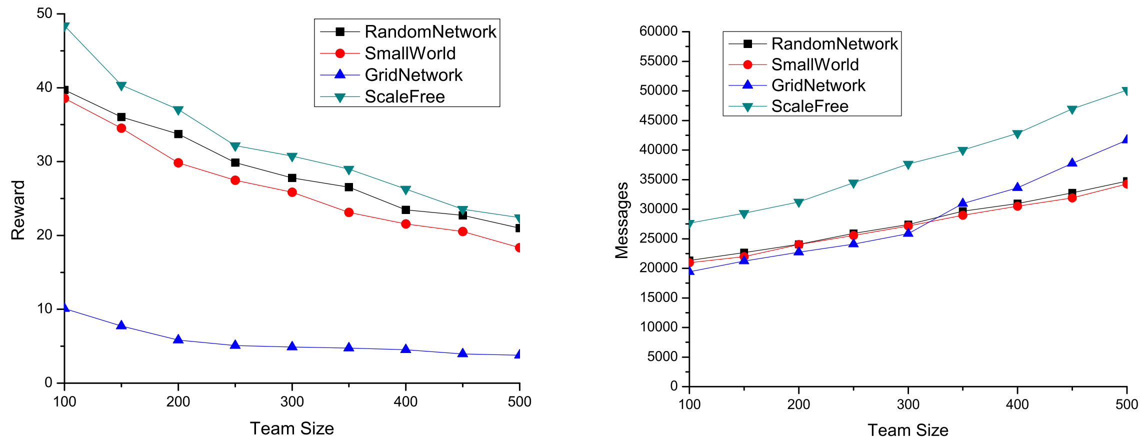


Figure 45: Application domain *Large-team coordination: social effects on different team sizes.

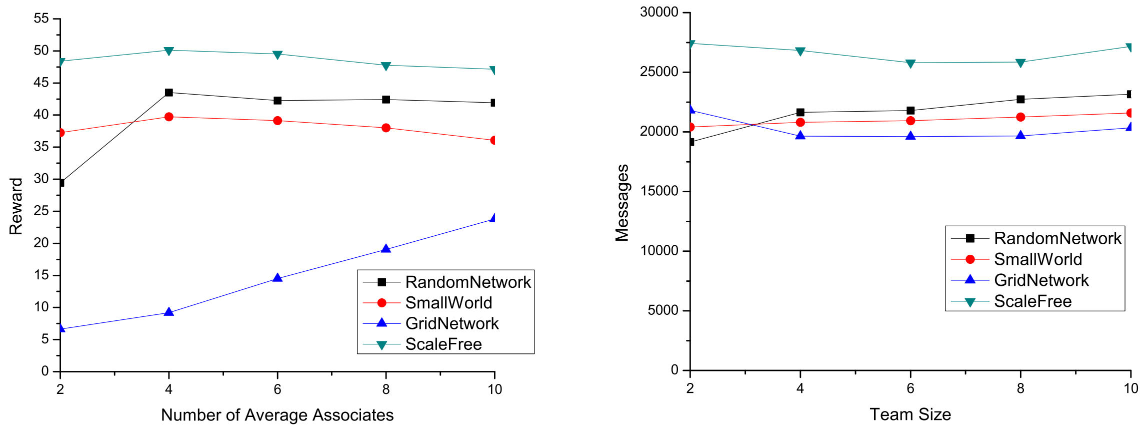


Figure 46: Application Domain *large-team coordination: social effects on different number of average associates.

10.0 CONCLUSIONS AND FUTURE WORK

In my thesis, I have presented a novel integrated token-based algorithm for scalable team coordination with several hundreds of agents. In this approach, I integrated three major types of coordination elements: information, resource and task into tokens, and converted this coordination problem into a communication decision problem. By utilizing relationships between tokens, intelligent routing algorithm efficiently routes tokens based on previous tokens and we were able to use the execution of one type of coordination algorithm to improve the performance of the others when the relevances between different types of tokens are defined. Although in this approach, agents' local decision to route tokens is based on the imprecise probability model from their previous incoming token, my theoretical analysis and pre-experiments have shown that this imprecise local decision model can make "good" decisions in large teams. The key of my research is the designing of intelligent routing algorithm that previously received tokens will help the receiver to infer whether the sender could benefit the team if a related token is received. Moreover, I have defined the team organization as associate network to limit agents' direct communication to a few of the others. This team organization model is efficient for token routing because more tokens are received from each associates and it is helpful for agents to build better models of their associates.

In chapter 9, I have setup my experiments in an abstracted coordination environment and verified my approach in three aspects: scalability, robustness and efficiency. To show the scalability, I applied my approach to different team size from 50 to 1000 and spanned in different application domains with different characters. By comparing with the baseline of random token movement and the partial coordination algorithms, I demonstrated that my integrated token-based algorithm outperformed them at any team size and any domains. By comparing with market-based approach which are only focused on maximizing overall utility at the expense of higher communications, token-algorithms are well focused on scalability and make the trade off between team performance and communication. In some cases, it gets comparable team utilities as auction with very limited communication overhead.

Imprecise definitions of relevance between tokens and plan conflictions are two major factors that influence the robustness of my approach. Efficiency of intelligent routing algorithm highly depends on the definition of relevance. I have demonstrated that my approach is robust to a high variance of imprecise relevance definition and its variance of utilities and communication cost are within a fixed interval. By defining the sub-teams for each active plans, my experiment results have shown most of the duplicated plans, which would badly influence the team performance, have been successfully solved.

As the third major contribution of my thesis research, I have shown different associate network topologies makes token-based coordination efficiency different in both theoretical and experimental measure. The small world effect and scale free effect enhanced the efficiency of token-based approach by making agents "closer" or introducing some degrees of centralized control to build better decision model for a few agents to pass tokens. Moreover, complete connections with any pairs of agents or agents maintaining many associates may deteriorate the intelligent routing algorithm.

This approach opens the possibility to develop a range of new executing applications of heterogeneous agents not possible with existing approaches. While the results presented in my thesis represent a step forward, they also point to significant challenges and exciting questions. I plan to address some of these issues in the future. Although the effect of the underlying associate network on the coordination is clearly important from my experiment results, how to make use the merits of the social network topologies and build intelligent associate network to foster the efficiency of my token-based algorithm most is a good topic for my future research.

This work represented a novel attempt at integrating coordination algorithms into a unified approach and showing how by working together the overall performance can be improved. However, the individual algorithms were designed without thought to future integration. A key question is whether knowing that I will be integrating the token algorithms allows us to build algorithms that work better with other algorithms. Finally, but critically, until now, I only demonstrated my algorithm in an abstracted simulator. I will use the token-based approach in more realistic domains to understand its utility in the real world. Specifically, my research group are currently developing Machinetta, a real distributed coordination system. I am going to integrated my approach into this system and applied in the large scale coordination applications for rescue response and unmanned aerial vehicle applications. The objective is to find more challenges in the real domains that critical to my token-based approach.

BIBLIOGRAPHY

[1] Token ring access method. In *IEEE. 802.5*.

[2] William Agassounon and Alcherio Martinoli. Efficiency and robustness of threshold-based distributed allocation algorithms in multiagent systems. In *Proceedings of AA-MAS'02*, 2002.

[3] R. Albert and A. Barabasi. Statistical mechanics of complex networks. *Review Modern Physics*, pages 74, 47, 2002.

[4] Marc Atkin, David Westbrook, and Paul Cohen. Capture the flag: Military simulation meets computer game. In *AAAI Spring Symposium on AI and Computer games*, 1999.

[5] N. Kalra B. Dias, R. Zlot and A. Stentz. Market-based multirobot coordination: A survey and analysis. Technical report, tech. report CMU-RI-TR-05-13, Robotics Institute, Carnegie Mellon University, 2005.

[6] Albert-Laszla Barabasi and Eric Bonabeau. Scale free networks. *Scientific American*, pages 60–69, May 2003.

[7] R. Beard, R. Frost, and W. Stirling. A hierarchical coordination scheme for satellite formation initialization. In *in AIAA Guidance, Navigation and Control Conference, Boston, MA.*, 1998.

[8] Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V. Goldman. Solving Transition Independent Decentralized Markov Decision Processes. *Journal of Artificial Intelligence Research*, 22:423–455, December 2004.

[9] Daniel S. Bernstein, Shiomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-00), Stanford, California*, pages 32–37, 2000.

[10] Avrim Blum and Merrick Furst. Fast planning through planning graph analysis. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, pages 1636–1642, 1995.

[11] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Theoretical Aspects of Rationality and Knowledge*, pages 195–201, 1996.

[12] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[13] M. E. Bratman. *Intention Plans and Practical Reason*. 1987.

[14] Johanna Bryson. Hierarchy and sequence vs. full parallelism in action selection. In *Intelligent Virtual Agents 2*, pages 113–125, 1999.

[15] Mark H. Burstein and David E. Diller. A framework for dynamic information flow in mixed-initiative human/agent organizations. *Applied Intelligence on Agents and Process Management*, 2004. Forthcoming.

[16] M. Campos, E. Bonabeau, G. Therauluz, and J.-L. Deneubourg. Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, 2001.

[17] J. Casper and R. Murphy. Workflow study on human-robot interaction in usar. In *International Conference on Robotics and Automation*, pages 1997–2003, 2002.

[18] Hans Chalupsky, Yolanda Gil, Craig A. Knoblock, Kristina Lerman, Jean Oh, David V. Pynadath, Thomas A. Russ, and Milind Tambe. Electric Elves: Agent technology for supporting human organizations. *AI Magazine*, 23(2):11–24, 2002.

[19] V. A. Cicirello and S. F. Smith. Wasp nests for self-configurable factories. In *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001.

[20] Richard Clark. *Uninhabited Combat Air Vehicles: Airpower by the people, for the people but not with the people*. Air University Press, 2000.

[21] S.E. Conry, K. Kuwabara, V.R. Lesser, and R.A. Meyer. Multistage Negotiation in Distributed Constraint Satisfaction. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6):1462–1477, November 1992.

[22] K. Decker and V. Lesser. Generalizing the Partial Global Planning Algorithm. *International Journal on Intelligent Cooperative Information Systems*, 1(2):319–346, June 1992.

[23] K. Decker, K. Sycara, A. Pannu, and M. Williamson. Designing behaviors for information agents. In *Procs. of the First International Conference on Autonomous Agents*, 1997.

[24] Vincent Decugis and Jacques Ferber. Action selection in an autonomous agent with a hierarchical distributed reactive planning architecture. In *Proceedings of the Second International Conference on Autonomous Agents*, 1998.

[25] Defense Science Board. Defense science board study on unmanned aerial vehicles and uninhabited combat aerial vehicles. Technical report, Office of the Under Secretary of Defense for Acquisition, Technology and Logistics, 2004.

[26] B. Dias and A. Stentz. Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination. Technical report, tech. report CMU-RI -TR-03-19, Robotics Institute, Carnegie Mellon University, 2003.

[27] Tara A. Estlin, Alexander Gray, Tobias Mann, Gregg Rabideau, Rebecca Castano, Steve Chien, and Eric Mjolsness. An integrated system for multi-rover scientific exploration. In *AAAI/IAAI*, pages 613–620, 1999.

[28] P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Int. Journal of Robotics and Autonomous Systems*, 24(3-4):159–182, 1998.

[29] Alessandro Farinelli, Paul Scerri, and Milind Tambe. Building large-scale robot systems: Distributed role assignment in dynamic, uncertain domains. In *Proceedings of Workshop on Representations and Approaches for Time-Critical Decentralized Resource, Role and Task Allocation*, 2003.

[30] S. Shaheen Fatima and Michael Wooldridge. Adaptive task resources allocation in multi-agent systems. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 537–544, 2001.

[31] R. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In *IJCAI*, pages 608–620, 1971.

[32] Brian P. Gerkey and Maja J Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, October 2002.

[33] Joseph Giampapa and Katia Sycara. Team oriented agent coordination in the RETSINA multi-agent system. In *Proceedings of Agents02*, 2002.

[34] Dani Goldberg, Vincent Cicirello, M Bernardine Dias, Reid Simmons, Stephen Smith, and Anthony (Tony) Stentz. Market-based multi-robot planning in a distributed layered architecture. In *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2003 International Workshop on Multi-Robot Systems*, volume 2, pages 27–38. Kluwer Academic Publishers, 2003.

[35] C. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multiagent systems. 2003.

[36] C. V. Goldman and S. Zilberstein. Mechanism design for communication in cooperative systems. In *Fifth Workshop on Game Theoretic and Decision Theoretic Agents*, 2003.

[37] Barbara Grosz and Sarit Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–358, 1996".

[38] Barbara Grosz and Sarit Kraus. The evolution of sharedplans. *Foundations and Theories of Rational Agencies, A. Rao and M. Wooldridge, eds*, 86:227–262, 1999.

[39] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored mdps, 2001.

[40] Y.-C. Ho. Team decision theory and information structures. In *Proceedings of the IEEE, vol. 68, no. 6, pp. 644–654.*, 1980.

[41] L. Hunsberger and B. Grosz. A combinatorial auction for collaborative planning, 2000.

[42] N. Jennings. The archon systems and its applications. Project Report, 1995.

[43] N. R. Jennings. Specification and implementation of a belief-desire-joint-intention architecture for collaborative problem solving. *Intl. Journal of Intelligent and Cooperative Information Systems*, 2(3):289–318, 1993.

[44] Nick Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75:195–240, 1995.

[45] Hiraoki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, , and Hitoshi Matsubara. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, Spring 1997.

[46] Hiroaki Kitano, Satoshi Tadokoro, Itsuki Noda, Hitoshi Matsubara, Tomoichi Takahashi, Atsushi Shinjoh, and Susumu Shimada. Robocup rescue: Searh and rescue in large-scale disasters as a domain for autonomous agents research. In *Proc. 1999 IEEE Intl. Conf. on Systems, Man and Cybernetics*, volume VI, pages 739–743, Tokyo, October 1999.

[47] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. NagendraPrasad, A. Raja., R. Vincent, P. Xuan, and X. Q. Zhang. Evolution of the gpgp/taems domain-independent coordination framework. *Autonomous Agents and Multi-Agent Systems*, 9(1), 2004.

[48] D. Li, K. Wong, Y. Hu, and A. Sayeed. Detection, classification, and tracking of targets, 2002.

[49] E. Liao, P. Scerri, and K. Sycara. A framework for very large teams. In *AAMAS'04 Workshop on Coalitions and Teams*, 2004.

[50] Michael L. Littman. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, pages 748–761, Providence, Rhode Island, 1997". AAAI Press / MIT Press.

[51] L. Lovasz. A decentralized approach to space deconflictionrandom walks on graphs: A survey. *in Combinatorics Bolyai Mathematical Society*, 1993.

[52] A. Cassandra M. Littman and L. Kaelbling. Learning policies for partially observable environments: scaling up. 1995.

[53] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS'04*, 2004.

[54] Roger Mailler, Victor Lesser, and Bryan Horling. Cooperative negotiation for soft real-time distributed resource allocation. In *Proceedings of AAMAS'03*, 2003.

[55] John Simmons Matthew E. Gaston and Marie desJardins. Agent-organized networks for dynamic team formation. In *2005 International Conference on Autonomous Agents and Multi-Agent Systems, Utrecht, Netherlands*, 2005.

[56] Amnon Meisels. Distributed constraints satisfaction. algorithms, performance, communication. In *Tenth International Conference on Principles and Practice of Constraint Programming*, page Tutorial, 2004.

[57] S. P. Meyn and R. L. Tweedie. Markov chains and stochastic stability. In *London: Springer-Verlag, ISBN 0-387-19832-6.*, 1993.

[58] S. Milgram. The small world problem. In *Psychology Today, 22, 61-67*, 1967.

[59] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proceedings of Autonomous Agents and Multi-Agent Systems*, 2003.

[60] R. Nair, T. Ito, M. Tambe, and S. Marsella. Task allocation in robocup rescue simulation domain. In *Proceedings of the International Symposium on RoboCup*, 2002.

[61] R. Nair, M. Tambe, and S. Marsella. Role allocation and reallocation in multiagent teams: Towards a practical analysis. In *Proceedings of the second International Joint conference on agents and multiagent systems (AAMAS)*, 2003.

[62] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings, 2003.

[63] M. E. J. Newman. The structure and function of complex networks. In *SIAM Review*, volume Vol. 45, No. 2, pages 167-256, 2003.

[64] Regis Vincent Paul Scerri and Roger Mailler. Comparing three approaches to large scale coordination. In *Proceedings of AAMAS'04 Workshop on Challenges in the Coordination of Large Scale MultiAgent Systems*, 2004.

[65] Barney Pell, Douglas E. Bernard, Steve A. Chien, Erann Gat, Nicola Muscettola, P. Pandurang Nayak, Michael D. Wagner, and Brian C. Williams. An autonomous

spacecraft agent prototype. In *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 253–261, New York, 1997. ACM Press.

[66] M. Prokopenko, M. Butler, and T. Howard. On emergence of scalable tactical and strategic behaviour. In Peter Stone, Tucker Balch, and Gerhard Kraetszchmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, Berlin, 2001. Springer Verlag. To appear.

[67] David Pynadath and Milind Tambe. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, 2002.

[68] David V. Pynadath and Milind Tambe. An automated teamwork infrastructure for heterogeneous software agents and humans. *Journal of Autonomous Agents and Multi-Agent Systems, Special Issue on Infrastructure and Requirements for Building Research Grade Multi-Agent Systems*, page to appear, 2002.

[69] D.V. Pynadath, M. Tambe, N. Chauvat, and L. Cavedon. Toward team-oriented programming. In *Intelligent Agents VI: Agent Theories, Architectures, and Languages*, pages 233–247, 1999.

[70] M. Tambe M. Yokoo R. Nair, P. Varakantham. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *AAAI*, 2005.

[71] Cesar Fernandez Carla Gomes Bhaskar Krishnamachari Bart Selman Magda Valls Ramon Bejar, Carmel Domshlak. Sensor networks and distributed csp: communication, computation and complexity. *Artif. Intell.*, 161(1-2):117–147, 2005.

[72] Paul Ranky. *An Introduction to Flexible Automation, Manufacturing and Assembly Cells and Systems in CIM (Computer Integrated Manufacturing), Methods, Tools and Case Studies*. CIMware, 1997.

[73] Anand Rao and Michael Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning(KR'91)*, Cambridge, MA, USA, 1991.

[74] Bharaneedharan Rathnasabapathy and Piotr Gmytrasiewicz. Formalizing multi-agent pomdp's in the context of network routing. In *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9*, page 301.1, 2003.

[75] C. Reynolds. Authoring autonomous characters. Invited Talk, Distinguished Lecture Series, Georgia Institute of Technology, Fall 1995.

[76] C. Rich and C. Sidner. COLLAGEN: When agents collaborate with people. In *Proceedings of the International Conference on Autonomous Agents (Agents'97)"*, 1997.

[77] Pedro Sander, Denis Peleshchuk, and Barabara Grosz. A scalable, distributed algorithm for efficient task allocation. In *Proceedings of AAMAS'02*, 2002.

[78] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. In *Artificial Intelligence, 135*, 2002.

[79] Owens S. Yu B. Scerri, P. and K. Sycara. A decentralized approach to space deconfliction. In *Proceedings of Fusion 2007 International Conference, Quebec City, Canada.*, 2007.

[80] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proceedings of Fourth International Joint Conference on Autonumous Agents and Multiagent Systems*, 2005.

[81] P. Scerri, E. Liao, Yang. Xu, M. Lewis, G. Lai, and K. Sycara. *Theory and Algorithms for Cooperative Systems*, chapter Coordinating very large groups of wide area search munitions. World Scientific Publishing, 2004.

[82] P. Scerri, D. Pynadath, and M. Tambe. Adjustable autonomy in real-world multi-agent environments. In *Proceedings of the Fifth International Conference on Autonomous Agents (Agents'01)*, pages 300–307, 2001.

[83] P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr, M Si, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *The Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003.

[84] P. Scerri, K. Sycara, and M Tambe. Adjustable autonomy in the context of coordination. In *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, 2004. Invited Paper.

[85] P. Scerri, Yang. Xu, E. Liao, J. Lai, and K. Sycara. Scaling teamwork to very large teams. In *Proceedings of AAMAS'04*, 2004.

[86] Jiaying Shen and Victor Lesser. Communication Management Using Abstraction in Distributed Bayesian Networks. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 622–629. ACM Press, 2006.

[87] K. Sycara and M. Lewis. *Team Cognition*, chapter Integrating Agents into Human Teams. Erlbaum Publishers, 2003.

[88] V. Guralnik T. Wagner and J. Phelps. A key-based coordination algorithm for dynamic readiness and repair service coordination. In *Proceedings of Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003.

[89] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[90] M. Tambe, D. Pynadath, C. Chauvat, A. Das, and G. Kaminka. Adaptive agent architectures for heterogeneous team members. In *Proceedings of ICMAS'2000*, pages 301–308, 2000.

[91] M. Tambe, D. Pynadath, and N. Chauvat. Building dynamic agent organizations in cyberspace. *IEEE Internet Computing*, 4(2):65–73, March 2000.

[92] Milind Tambe. Agent architectures for flexible, practical teamwork. *National Conference on AI (AAAI97)*, pages 22–28, 1997.

[93] Milind Tambe and Weixiong Zhang. Towards flexible teamwork in persistent teams: Extended report. *Autonomous Agents and Multi-Agent Systems*, 3(2):159–183, 2000.

[94] A. S. Tanenbaum. Computer networks. In *Prentice hall*.

[95] G. Tidhar, A.S. Rao, and E.A. Sonenberg. Guided team selection. In *Proceedings of the Second International Conference on Multi-Agent Systems*, 1996.

[96] J. Travers and S. Milgram. An experimental study of the small world problem. In *Sociometry, Vol32 425¨C443*, 1969.

[97] Thomas Wagner, Anita Raja, and Victor Lesser. Modeling Uncertainty and its Implications to Sophisticated Control in TAEMS Agents. *Autonomous Agents and Multi-Agent Systems*, 13(3):235–292, November 2006.

[98] Thomas Wagner, Jonathan Shapiro, Ping Xuan, and Victor Lesser. Multi-Level Conflict in Multi-Agent Systems. *Proceedings of the AAAI-99 Workshop on Negotiation in MAS*, pages 50–55, 1999.

[99] Duncan Watts. Beyond the small world. *Chapter 4 of D.J. Watts, Six Degrees: The Science of a Connected Age*, 2002.

[100] Duncan Watts and Steven Strogatz. Collective dynamics of small world networks. *Nature*, 393:440–442, 1998.

[101] D. J. White. Markov decision processes. In *West Sussex, England*.

[102] Y. Xu, M. Lewis, K. Sycara, and P. Scerri. Information sharing in very large teams. In *In AAMAS'04 Workshop on Challenges in Coordination of Large Scale MultiAgent Systems*, 2004.

[103] Yang Xu, Paul Scerri, Katia Sycara, and Michael Lewis. Comparing market and token-based coordination. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1113–1115. ACM Press, 2006.

[104] Yang Xu, Paul Scerri, Bin Yu, Steven Okamoto, Katia Sycara, and Michael Lewis. An integrated token-based algorithm for scalable coordination. In *Proceedings of Fourth International Joint Conference on Autonumous Agents and Multiagent Systems*, 2005.

[105] P. Xuan, V. Lesser, and S. Zilberstein. Communication decisions in multi-agent cooperation: Model and experiments. In *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001.

[106] J. Yang. *Co-ordination Based Structured Parallel Programming.* PhD thesis, Imperial College of Science, Technology and Medicine, University of London, 1998.

[107] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *Knowledge and Data Engineering*, 10(5):673–685, 1998.

[108] Makoto Yokoo and Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, 2000.

[109] Anita Raja XiaoQin Zhang and Barbara Lerner. Integrating High-Level and Detailed Agent Coordination into a Layered Architecture. *The workshop on Infrastructure for Scalable Multi-Agent Systems, Agent's 2000. Also available as UMass Computer Science Technical Report 1999-029.*, June 2000.

[110] Roie Zivan and Amnon Meisels. Concurrent search for distributed csps. *Artifical Intelligence*, 170(4):440–461, 2006.