6-2017

INVESTIGATING AGENT AND TASK OPENNESS IN ADHOC TEAM FORMATION

Bin Chen
University of Nebraska - Lincoln, franky_chen2008@hotmail.com

INVESTIGATING AGENT AND TASK OPENNESS IN ADHOC TEAM FORMATION

by

Bin Chen

A THESIS

Presented to the Faculty of

The Graduated College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Leen-Kiat Soh

Lincoln, Nebraska

June, 2017

INVESTIGATING AGENT AND TASK OPENNESS IN ADHOC TEAM FORMATION

Bin Chen, M.S

University of Nebraska, 2017

Advisor: Leen-Kiat Soh

When deciding which ad hoc team to join, agents are often required to consider rewards from accomplishing tasks as well as potential benefits from learning when working with others, when solving tasks. We argue that, in order to decide when to learn or when to solve task, agents have to consider the existing agents' capabilities and tasks available in the environment, and thus agents have to consider agent and task openness—the rate of new, previously unknown agents (and tasks) that are introduced into the environment. We further assume that agents evolve their capabilities intrinsically through learning by observation or learning by doing when working in a team. Thus, an agent will need to consider which task to do or which team to join would provide the best situation for such learning to occur. In this thesis, we develop an auction-based multiagent simulation framework, a mechanism to simulate openness in our environment, and conduct comprehensive experiments to investigate the impact of agent and task openness. We propose several agent task selection strategies to leverage the environmental openness. Furthermore, we present a multiagent solution for agent-based collaborative human task assignment when finding suitable tasks for users in complex environments is made especially challenging by agent openness and task openness. Using an auction-based protocol to fairly assign tasks, software agents model uncertainty in the outcomes

of bids caused by openness, then acquire tasks for people that maximize both the user's utility gain and learning opportunities for human users (who improve their abilities to accomplish future tasks through learning by experience and by observing more capable humans). Experimental results demonstrate the effects of agent and task openness on collaborative task assignment, the benefits of reasoning about openness, and the value of non-myopically choosing tasks to help people improve their abilities for uncertain future tasks.

ACKNOWLEDGEMENTS

First, I would like to sincerely express my gratitude to my advisor Dr. Leen-Kiat Soh. It is his continuous support, meaningful feedback, insightful suggestions, and professional guidance that made this research successful. I am so thankful that I have such an adviser, a mentor, and a friend in my life.

Second, I thank my colleagues for their willingness to have meaningful discussions and critiques which improved my work. I especially thank Xi Chen for his collaborating in developing the simulator for this research and Dr. Adam Eck for helping with the formulation of the theory behind the agent models.

Third, I acknowledge the resources available at the University of Nebraska-Lincoln that helped me completing this research, including research papers, journals and the super computers hosted by Holland Computing Center that made available to me.

Last but not the least, I thank my wonderful wife Xiao Liang for her love, support, and sacrifice which made me carry out my research.

Table of Contents

TABLE OF FIGURES

Table of Tables

Table of Algorithms

Chapter 1: Introduction

1.1 Problem

Intelligent agents are capable of sensing the environment, making autonomous decisions which in turn influence the environment. A multiagent system consists of such agents that work together cooperatively or competitively towards a common goal. Multiagent systems provide a strong platform for examining coalition formation and member interaction. Agents can mirror the operation of people in actual groups. Modeling how agents form coalitions within the broader group has been an active area in multiagent systems (Caillou, Aknine, & Pinson, 2002; Onn Shehory & Kraus, 1998; Soh & Tsatsoulis, 2002). However, the most relevant subarea concerns modeling cooperative multiagent systems where agents learn to coordinate with their cooperative team members *without* having any prior collaboration experience with them (Stone, Kaminka, & Rosenschein, 2010b), has not been extensively studied. In such a setting, different agents may have different capabilities and tasks may need varieties of capabilities to be completed. Furthermore, agents may be programed by others, may or may not be able to communicate, and teammates are likely sub-optimal. Due to the uncertainty and dynamic changes of the environment, the ad hoc teams formed may result in inefficient or ineffective task solutions.

There are many aspects of ad hoc team formation that have been studied, focusing on learning, leading, and dealing with uncertainties in agent behavior (Agmon, et al., 2014; Barrett et al., 2012; Jumadinova et al., 2014; Stone, Gan, et al., 2010; Stone, Kaminka, et al., 2010; Wooldridge, 2009). For example, Stone, Kaminka, et al., (2010b)

proposed ad hoc teams where agents work together without pre-coordination in highly uncertain and dynamic environments. Stone, Gan, et al., (2010) presented a probabilistic hill-climbing-based algorithm that allows autonomous agents with heterogeneous expertise to learn how to coordinate in coalitions that contain unknown agents to solve collaborative tasks. These research approaches capture a number of the necessary aspects (e.g., unknown teammates, heterogeneous expertise, and task solution that requires collaboration) of our coalition formation problem.

However, the emphasis of such ad hoc team play problems is not on how the agent coalitions themselves form. As we try to study team formation in certain agents, like human, we need to consider several factors like how human learn from working in a team as well as observing a teammate. Research done so far, while considering learning (Barrentt et al. 2012), has not considered the learning that is present when agents—such as humans—work together in a team. For example, when human agents work together, it is inevitable that they learn from each other, and occasionally they teach each other. Indeed, human agents do learn and evolve when they interact and work in a team through time. Through learning, agents can improve their capabilities so that they can do things better next time and improve the efficiency of the entire system. It is the learning that makes agents evolve in a dynamic complex system and adapt to the changes in the environment. In ad hoc team formation, while prior knowledge of a potential teammate is not available, it is still possible for an agent to model the types of agents and tasks likely to be in the environment, and to assume that learning is inevitable when working together. Such consideration and assumption will influence how agents form ad hoc teams—in how each decides to join an ad hoc team to help solve a task. Thus, it is

necessary to consider learning when agents work together and its impact in subsequent tasks.

Furthermore, a key question to ad hoc team formation is how agents should decide on which teams to join when taking into account the potential rewards or utility of learning while on a team. In a way, if learning consumes resources or its effectiveness might come at the cost of the overall rewards for solving the task, then there is a tradeoff. That is, an agent would have to trade off between combined reward resulting from optimizing on task rewards and that resulting from optimizing on learning. Should an agent focus on learning now and sacrifice task rewards? Or should it focus on getting paid as much as possible now with the task rewards and worry about learning later? In an ad hoc environment where an agent has little or no knowledge about each individual potential teammate, how should such an agent leverage what it can model of the environment to help make this decision?

We see that there are two types of openness from a multiagent viewpoint, extending the concepts from what have been proposed by Jumadinova et al., (2014). First, task openness refers to the rate of new, previously unseen tasks that are introduced into the environment. Second, agent openness refers to the rate of new, previously unknown agents that are introduced into the environment, while known agents exit the environment. For example, an agent whose particular capability is low may choose to join a team with a good opportunity to learn about this capability from other teammates even when the direct rewards of completing this task is low. Thus, if the degree of agent openness is high, such that different agents enter the environment and exit from it very often, then the likelihood to work with the same agent/agent type to learn about a

particular capability would be low. So, it might be prudent for the agent to lean towards joining a team to learn from the particular agent/agent type sooner than later. Also, if the task openness is high, such that different tasks appear and disappear from the environment very often, then the likelihood of encountering the same task/task type again would be low, then agents do not have to spend time, effort, and resource to learn to solve a particular task/task type—say, a difficult one—if the task/task type would not likely appear again in the future. In that case, an agent might not care too much about learning to solve that task/task type, and instead aim for getting more direct rewards sooner.

Agent openness and task openness, as well as the fact that agents will learn and evolve, make our open system a challenging system and yet very different from the traditional dynamic systems. In traditional dynamic system, agents may be faulty and go offline, then they may or may not come back to the system. In our open system, the agent openness causes a set of agents changes, making old agents leave the environment and disappear forever as well as brand new agents, which the existing agents have never seen before, entering the system and thus forcing the existing agents to have to learn something new about theses brand new agents. The agents who leave the system take the expertise out of the system while the brand new agents who enter the system bring new expertise into the system. In such a system, agents constantly have to work with new different agents in general, which sets our open system apart from the regular dynamic system. The injection of new agents into the environment causes changes in our agents' reasoning in two ways: (1) when an agent reasons or learns, not only it has to think about agent leaving, but also new agents entering; and (2) when an agent reasons, learns, or

acts, it has to work with new agents and it is impacted directly by its experience with these new agents and the loss of existing agents from its environment.

Furthermore, according to task openness, tasks are changing over time. Old tasks leave the system and new tasks come into the system. Agents never know for certainty in advance what they need to do and what expertise they need to learn to benefit them from completing the future tasks. The task openness has an interesting correlation with learning, since agents want to learn to get better in the future, but they do not know for certainty what tasks are going to be available. This forces agents to model the environment and make decisions about what to learn and from whom to learn.

1.2 Motivation

Intelligent agents and multiagent systems have been used in a wide variety of applications to support human activities and decision making.

One particular problem that agents are well suited to assist human users with is collaborative task assignment, where there exist a set of human users and a set of tasks that require multiple people to combine their individual skills and expertise to work together towards a common, temporary goal, earning each participant a share of a joint reward if the task is accomplished successfully. In such a problem, a multiagent solution is advantageous because agents representing individual human users can first model the abilities of their assigned users, then find and acquire tasks that best benefit their users, while at the same time fairly allocate tasks across all users so that the overall system also benefits. For example, agent-based human collaborative task assignment could be used to (1) form temporary teams of freelance workers (e.g., independent software developers

or artists) to satisfy contracts from companies lacking the internal expertise to accomplish tasks (e.g., developing particular pieces of software or graphic design), (2) combine the expertise and skills of office workers across divisions within large companies to accomplish tasks needed by the company, or (3) further improve matching students to peer-based learning tasks in computer-aided education.

However, collaborative task assignment becomes much more challenging within dynamic, open environments where the system itself changes due to entities coming and going over time. In particular, we consider two types of openness affecting the collaborative task assignment problem. First, agent openness occurs whenever the set of human agents changes as people join and leave the environment over time. This causes expertise and skills needed to accomplish tasks to become more or less prevalent, affecting the ability of software agents to find suitable people to accomplish each task. For instance, if an expert and skilled person leaves the environment, then tasks that could be successfully accomplished in the past might not be possible anymore. Second, task openness occurs whenever the set of collaborative tasks changes: both new tasks requiring different expertise and skills appear and older tasks disappear over time. People specializing in certain types of tasks might need to adapt what they work on if those tasks disappear, while other people who had difficulty contributing might become more useful as new tasks related to their expertise and skills appear.

Both types of openness cause uncertainty within the collaborative task assignment problem, as software agents do not know which tasks might be successfully accomplished now or in the future due to fluctuations in both the set of people needed to complete tasks, as well as the set of tasks itself. Given that there might be multiple tasks

each person could contribute to at any point in time, yet a person can only contribute to one task at a time, openness makes the problem of selecting appropriate tasks for human users more difficult for software agents.

1.3 Proposed Solution

Our work uses a learner-driven approach for ad-hoc collaboration in a multi-agent task execution scenario. In our scenario, tasks can be broken down into different subtasks, each requiring certain expertise or capability to be completed. Meanwhile, each agent can improve its capabilities either by performing the subtask or observing other members solving the subtask in the team. Agents are autonomous. Consequently, each agent tries to improve its chance for getting selected in a task by improving the quality of its capabilities that maybe needed for future task.

First, we have developed an ad hoc team formation framework that takes into account learning and task solving under varying degrees of environmental openness. The learning involved is based on "learning by observation" and "learning by doing" modeling learning theory on the zone of proximal distance. An additional emphasis here is about how an agent can choose a subtask to do such that joining a team to help complete an overall task allows the agent to position itself to gain from learning, from doing the subtask and from observing others working in the team. Furthermore, we have devised mechanisms to simulate agent and task openness. Running simulations of this framework, we were able to study various effects of considering agent openness (AO) and task openness (TO) in ad-hoc team formation.

Second, we have applied our ad hoc team formation framework to an agent-based collaborative human task assignment problem. We have particularly addressed agent openness and task openness in this problem. We have further modeled human learning by doing and by observation, and incorporated these into the agent's reasoning about how to acquire tasks for its user. Our solution develops an approach for modeling and learning unmeasurable uncertainty caused by environment openness to guide its decision making in maximizing human user reward and learning gains over sequences of tasks.

1.4 Contributions

First, we have developed an auction-based multiagent simulation framework, which is a mechanism to simulate openness in our environment, and have conducted comprehensive experiments. We have developed a Java based simulation package for our framework, which allows researchers to conduct extensive experiments to study ad hoc team formation problem. Chapter 5 talks about this work in detail.

Second, we have established the importance of agent openness and task openness, gained insights into the relationship between the two factors, and investigated the effectiveness of several openness-based task selection strategies. In addition, we have identified several key next steps to continue with this line of research. Chapter 2 details such work. Chen et al., (2015) has published this work on the Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems as extended abstract.

Third, we have studied an agent-based collaborative human task assignment problem, which is a direct application of ad hoc team formation problem in open system.

We have developed solutions for agents to maximize their users' rewards and learning gains over sequence of tasks. Chapter 4 talks about this work. Chen et al., (2016) has published this work on the Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems as extended abstract.

1.5 Overview

The rest of the chapters are organized as follows. First, Chapter 2 summarizes the related work in ad hoc team formation research. Chapter 3 discusses the investigations we have done in detail in ad hoc team formation in open system, including our auction based framework, simulation of openness, our proposed algorithms, as well as the empirical results of simulations and future work. Chapter 4 discusses how we applied the agent-based solution to collective human task assignment problem in detail, including the human learning model, the methodologies we used, the empirical results, and future work. Chapter 5 gives the details of our test bed and Chapter 6 concludes our work and identifies the future work.

Chapter 2: Background and Related Work

In this chapter, we first discuss the background and related work for multiagent ad-hoc team formation (Section 2.1). Then, we describe the background and related work for multiagent task allocation problem (Section 2.2), which is mentioned in the first chapter as a direct application of our ad hoc team formation framework.

Wooldridge & Jennings (1995) described an **agent** as a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives. An agent typically senses the environment and has some predefined actions that can be executed to affect the environment. Shoham & Leyton-Brown (2008) defined a **multiagent system** as one that consists of a number of agents, which interact with one another, typically by exchanging messages through some computer network infrastructure. In such a system, agents need to interact with each other, hence they need to cooperate, coordinate, and negotiate.

Team formation is the problem of selecting the best possible team to accomplish a certain goal, given limited resources. In the traditional model, certain skills are necessary to accomplish a task, and we must select a team that has all the necessary skills with the maximum expected value (Marcolino, Jiang, & Tambe, 2013). In such a setting, tasks usually need multiple agents' actions (cooperative actions) to be completed. Hence agents need to cooperate to form teams to perform collective actions to complete the task. Our ad hoc team formation framework allows agents to select best tasks to their interests and form a team to complete the tasks. We will further elaborate this later in this chapter.

A multiagent environment can have different properties, as classified by Russell and Norving (1995 p.46). An environment can be deterministic, in which the actions has a guaranteed effect, or it can be non-deterministic. Also, an environment can be static or dynamic. A static environment is the environment that can be assumed to be unchanged except by the actions of the agent, while dynamic environment changes without agent's action and the changes is beyond the agent's control. In addition, an environment can be discrete or continuous. An environment is discrete if there are a fixed, finite number of actions and percepts in it. We can see that an environment can be complex. Hewitt (1986) referred to the environment that is inaccessible, non-deterministic, dynamic, and continuous as open. Our ad hoc team formation framework simulates such an open environment. In such environments, tasks can appear and disappear without notice, and agents can come and go as they please.

In this thesis, we are interested in the investigation of the impact of agent and task openness in ad hoc team formation in complex environments.

2.1 Multiagent Ad Hoc Team Formation

The team formation task is to select the best possible team to accomplish a certain goal. Existing team formation approaches often assume that the agents capabilities are known (e.g., Zhang & Parker (2012)). However, there are many real-world scenarios where different agents or robots with various of capabilities do not know each other, yet they have to coordinate and work in a team to complete a task or to meet a temporary goal. One of the scenarios is the disaster search and rescue scenario. When the disaster occurs, rescue teams rush into the areas that need help to provide assistance. Many search

and rescue robots are brought to the scene. Some of them are deployed to the site to complete some difficult or dangerous tasks. Many of these robots have not collaborated before, hence their capabilities are unknown to each other. Some of the robots are designed to work well with other types of robots, while some of them may not even have the ability to coordinate with each other. As a result of this, team strategies cannot be determined a priori. In such an ad hoc team formation problem, where team members have not collaborated before and they assume no prior knowledge of each other, selecting the agents/robots to form an optimal team is a challenging task.

Stone, Kaminka, & Rosenschein (2010a) raised a question to challenge the AI community to create an autonomous agent that is able to efficiently and robustly collaborate with previously unknown teammates on tasks to which they are all individually capable of contributing as team members. As we expect agents to be capable of performing complex tasks and representing real world scenarios, there will be a need to develop agents which can function with autonomy, for longer periods of time, interacting with older legacy agents, and agents with different communication protocols or world models. This requires agents that are capable of adapting with respect to other agents' behavior.

Stone, Gan, et al. (2010) introduced a problem which is formulated as multi-armed bandit (MAB) problem with a teacher and learner agent. In this problem teacher and learner agents try to optimize a team goal (collect maximum number of cans). The teacher agent has to decide on either optimizing its own utility (collect higher number of cans itself), or going for a sub-optimal option in order to teach the learner agent. This MAB problem only considered the case that the remaining arm pulls are finite. Later on,

Barrett & Stone (2011) extended the result with the consideration of an infinite number of arm pulls with discounted rewards. One key factor in their work is that the teacher and learner agents are always present in the environment and do not leave. If, on the other hand, the agents can leave and new agents can enter the environment, there could be very different implications. Based on how frequently agents leave (or new agents enter) the environment, teaching might have to be done more frequently, less frequently, or even none at all. For example, if an agent is only in the environment for a very short time, then it could be better for the teacher agent to not teach, and instead improve its own utility as it does not make sense for the teacher agent to teach, when the learner agent might leave quickly, without staying long enough to implement and improve the team's utility with what it has learned. It stands to reason that teaching frequently would be more beneficial only if the learning agent remains in the environment for a longer period of time, actually reaping the benefits of the new knowledge it has gained. Also, since the tasks in the environment are fixed, there is a guarantee on available tasks, and there are benefits of learning. Our consideration is that of an open environment, where task openness is considered, e.g., a task might have to be done frequently, or it could be a one-time task only. If the probability of certain task appearing in the system is more frequent, teaching other agents to solve those tasks would be beneficial. If not, then teaching would not be necessary and the knowledge gained to solve that particular task would likely not be used. This means, the decision to teach or not teach, would benefit from taking this factor into account, thus calling for the analysis on task and agent openness.

Stone, Kaminka, & Rosenschein (2010) introduced a game-theoretic formulation problem in multiagent teamwork. The authors studied a two-player game where one

intelligent agent interacts with an old legacy agent that can respond by selecting its best response to a fixed history of actions. An algorithm for finding optimal sequence of actions is given for the intelligent agent to find the sequence of actions which will lead the old legacy agent to achieve the best joint long-term payoff. This work has been extended to using a single agent to lead multiple teammates to maximize the payoff through a series of joint optimal actions (Agmon & Stone, 2011). This work is considered ad hoc team formation by the authors since there are different types of agents involved (old agents and the new intelligent agents) and there are no direct communications between them and they never worked together before. In (Stone, Kaminka, & Rosenschein, 2010), the intelligent agent knows the full action policy of the old legacy agent but the old legacy agent assumes no knowledge of the intelligent agent. Though the two agents do not have direct communication nor they have prior collaboration, this setting is not purely ad hoc in terms of the amount of information that agents assume of their peers. What happens if both agents have absolutely no prior knowledge of each other? In this case, the intelligent agent must observe its peers to learn their action policy. What if the observed agents disappear? If the observed agents no longer appear in the environment, then all the learning effort made by the observing agent would be wasted. This work focused on how the new intelligent agent leads the old agent through joint actions to achieve maximum long-term goal instead of focusing on the team formation itself. In contrast, our work assumes neither prior knowledge of agents nor the number of agents available in the environment. We focus on the problem of ***how ad hoc teams should be formed*** to complete tasks so that the whole system can benefit in an open environment where both agents and tasks can come and go at any moment.

Another type of work in ad hoc team formation has been done by Wu, Zilberstein, & Chen (2011). With unknown teammates but the system states and joint actions being fully observable, Wu, Zilberstein, & Chen (2011) proposed an online planning algorithm that can be used by ad hoc agents to maximize the team's joint reward by optimizing the joint actions of the team. Their approach is based on constructing and solving a series of stage games and then using biased adaptive play to choose actions. The algorithm proposed combining the advantages of biased adaptive play and UCT (Monte-Carlo tree search). In their work, planning is treated as an optimization problem in the joint policy space, which is constrained by the limited capabilities of teammates. The authors focused on the type of ad hoc teams in which a target agent knows the number of teammates as well as a set of their feasible actions, also the system state and the joint action played at each step are fully observable by the agent. In this setting, the target agent must reason about the past action sequences of its teammates online, learn from these interactions, and adapt its actions to its teammates. However, unlike our research, their work did not consider the learning capabilities of ad hoc teammates and assumed the domain is known, but make no assumptions of the behavior of teammates (teammates can be rational, irrational or in between). In our work, agent does enhance its capabilities while carrying out tasks in a team. Another key difference is along the level of openness in the environment. In their work, the tasks are fixed. More specifically, the agents form a team to do one task only. Their work is focused on how to coordinate well to accomplish the task, while our work supports the possibility of agents re-forming teams to do other tasks. Furthermore, in their work, the agents are fixed. No new agents would join the team and no team members would leave the team.

In Barrett, Stone, & Kraus (2011), the focus was on how ad hoc agents can perform, especially in the pursuit domain, where the agents are predators, trying to capture a prey. The actions that the agents perform in this environment are to capture the prey. An ad hoc agent in this setting has to model its teammates and choose best response to better suit the objective of the team, which is to catch the prey. There is an element of learning in the scenario, but this is limited to just on that action of capturing the prey. But ad hoc agents might be required to perform multitude of tasks, requiring different types of skills, thereby making it beneficial for them to learn multiple skills. This consideration of learning multiple skills is not made in Barrett et al., (2011) as those agents do not perform multiple type of tasks, but only a single type of task. Also, the teams in the scenario described in Barrett et al., (2011) are "static", i.e. agents do not leave or enter the environment. The question we want to answer is, what might happen if agents can come and go as they please? For example, if a predator is replaced by a new predator, it would require other teammates to learn about the new predator teammate. Indeed, this dynamism in the environment motivates our research towards analyzing how the performance of teams is affected by the introduction of open environment in terms of tasks and teammates.

2.2 Multiagent Task Allocation

As mentioned in Chapter 1, our research in ad hoc team formation in open environment has many applications. One of the most related applications is collaborative human task assignment. In real world applications, there are many situations that the environment is open with respect to both workers and tasks. For example, when forming

temporary teams of freelance programmers to work on contracts, the availabilities of freelance programmers in the job market change over time. During an economy boom, the market is very attractive such that many skilled freelance programmers are drawn into the market. However, as soon as any of them is committed to a job, he or she will be tied up with that job and not available in the market for a certain period of time (assuming that only one job per programmer at any given time). Meanwhile, it is also possible that the boom evolves faster than the capabilities of programmers such that jobs might not find sufficient programmers to fill them because of the freelance workforce simply does not have enough capable programmers in certain skills. In an economic recession, the market becomes not that attractive, many freelance programmers are leaving the market while some of them who are willing to work at a relative lower pay scale stills remains in the market. In addition, the projects/tasks also vary with the market change, hence different skills are needed to meet the changing market needs. This scenario demonstrates the dynamism of the real-world situation. This dynamism is represented in the characteristics of our open system in terms of agent and task openness. We see that the study in the impact of openness in such open environment in ad hoc team formation can benefit the real world in many ways.

In fact, intelligent agents and multiagent systems have been used in wide variety of application to support human activities and decision making. For instance, there are autonomous personal assistants that support their users in carrying out tasks, managing schedules, and so forth. For example, Chalupsky et al. (2002) and Tambe et al. (2008) described Electric Elves that helped humans in accomplishing organizational activities, such as rescheduling meetings, selecting presenters for research meetings, tracking

people's locations, and organizing lunch meetings. Myers et al. (2007) described a system that relieved the user of routine tasks and intervened in situations where cognitive overload leads to oversights or mistakes by the user. Berry et al. (2006) described a personalized agent called PTIME for time management and meeting scheduling as part of a larger assistive agent system called CALO. There are also collaboration support systems aimed at identifying for human users other human users to help with problem solving, teamwork, and learning. For example, Vassileva et al. (2015) described PHelpS that helped workers find appropriate helpers among their peers when they were encountering problems while interacting with their database, and I-Help that matched students with their peer helpers for university courses. Khandaker et al. (2011) described computer-supported collaborative learning applications called I-MINDS and ClassroomWiki to form optimal student teams based on students' tracked and modeled behaviors. Finally, Sklar and Richards (2006) pointed out, in addition to peer learning agents, that there were also pedagogical agents and demonstrating agents used in human learning systems. Pedagogical agents (Chalupsky et al., 2002) are designed to facilitate learner motivation and learning. They act as tutors and model student learner profiles and the current state of knowledge to customize their interactions accordingly.

In recent years, intelligent agents are widely used in our lives to work together with humans to accomplish certain tasks (Maes, 1994), some systems have humans working as information collector and information processor along with autonomous software agents within the systems (Kamar, Gal, & Grosz, 2013; Manson & O'Neill, 2007). Some systems let the agents pass information-processing tasks to the human, and then collect and aggregate the results (Ahn et al., 2008). The relationship between humans and

machines/agents has been changed. Humans and agents now have more and more flexible social interactions. Jennings et al., (2014) defined this emerging class of systems/teams as human-agent collectives (HACs). In many cases, humans are playing the major role while agents are playing the supporting role to make suggestions while in some cases agents are playing the major role and humans are play the supporting role. For instance, the automatic parking systems on some of the newer cars allow the computer to make decisions on whether the parking space is big enough to park the car or not and the computer takes over the steering wheel, leaving the driver to only control the breaks. Another example would be Tesla's autopilot. The system offers auto steering, and adaptive cruise control, which allows the car to steer and keep a safe distance between the car in front. Human in this case only takes over when some corrections are needed in the rare case when the system cannot fully sense the environment. The HACs system/team allows agents and humans to interact/engage in flexible relationships to achieve a common goal. Flexible relationships mean sometimes humans are in control or take the lead, sometimes the computers do. The relationship between humans and computers can change in a dynamic way. Our human task assignment system is similar to HACs to some extent, but the relationships between humans and its assigned agents are fixed. Our system allows agents to interact with people to discover their preferences, skills, and expertise, then find suitable tasks that maximize both the user's utility gain and learning opportunities in complex environments by modeling uncertainty in the outcomes of bids caused by openness. To simplify the complicity of our system, we assume the users' abilities are accessed by experts and represented by numeric value between 0 to 1, where 0 means no ability and 1 means expert ability. We further assume that the tasks obtained

by agents are all completed by its human users in a fixed amount of time (1 tick), the failure of completing the tasks will be considered in the future work.

Chapter 3: Investigation on Agent and Task Openness

3.1 Introduction

Many aspects of ad hoc team formation have been studied, focusing on learning, leading, and dealing with uncertainties in agent behavior (Agmon, et al., 2014; Barrett et al., 2012; Jumadinova et al., 2014; Stone, Gan, et al., 2010; Stone, Kaminka, et al., 2010; Wooldridge, 2009). For example, Stone, Kaminka, et al. (2010b) proposed ad hoc teams where agents work together without pre-coordination in highly uncertain and dynamic environments. Stone, Gan, et al. (2010) presented a probabilistic hill-climbing-based algorithm that allows autonomous agents with heterogeneous expertise to learn how to coordinate in coalitions that contain unknown agents to solve collaborative tasks.

But as we try to study team formation in certain agents, like human, we need to consider several factors like how human learn from working in a team as well as observing a teammate. Research done so far, while considering learning (Barrentt et al. 2012), has not considered the learning that is present when agents—such as humans—work together in a team. For example, when human agents work together, it is inevitable that they learn from each other, and occasionally they teach each other. Indeed, human agents do learn and evolve when they interact and work in a team through time. Through learning, agents can improve their capabilities so that they can do things better next time and improve the efficiency of the entire system. In ad hoc team formation, while prior knowledge of a potential teammate is not available, it is still possible for an agent to model the types of agents and tasks likely to be in the environment, and to assume that learning is inevitable when working together. Such consideration and assumption will

influence how agents form ad hoc teams—in how each decides to join an ad hoc team to help solve a task. Thus, it is necessary to consider learning when agents work together and its impact in subsequent tasks.

Furthermore, a key question to ad hoc team formation is how agents should decide on which teams to join when taking into account the potential rewards or utility of learning while on a team. In a way, if learning consumes resources or its effectiveness might come at the cost of the overall rewards for solving the task, then there is a tradeoff. That is, an agent would have to tradeoff between combined reward resulting from optimizing on task rewards and that resulting from optimizing on learning. Should an agent focus on learning now and sacrifice on task rewards? Or should it focus on getting paid as much as possible now with the task rewards and worry about learning later? In an ad hoc environment where an agent has little or no knowledge about each individual potential teammate, how should such an agent leverage what it can model of the environment to help make this decision?

We see that there are two types of openness from a multiagent viewpoint, extending the concepts from what have been proposed by Jumadinova et al. (2014). First, task openness refers to the rate of new, previously unseen tasks that are introduced into the environment. Second, agent openness refers to the rate of new, previously unknown agents that are introduced into the environment, while known agents exit the environment. For example, an agent whose particular capability is low may choose to join a team with a good opportunity to learn about this capability from other teammates even when the direct rewards of completing this task is low. Thus, if the degree of agent openness is high, such that different agents enter the environment and exit from it very

often, then the likelihood to work with the same agent/agent type to learn about a particular capability would be low. So, it might be prudent for the agent to lean towards joining a team to learn from the particular agent/agent type sooner than later. Also, if the task openness is high, such that different tasks appear and disappear from the environment very often, then the likelihood of encountering the same task/task type again would be low, then agents do not have to spend time, effort, and resource to learn to solve a particular task/task type—say, a difficult one—if the task/task type would not likely appear again in the future. In that case, an agent might not care too much about learning to solve that task/task type, and instead aim for getting more direct rewards sooner.

Our work in this investigation uses a learner-driven approach for ad-hoc collaboration in a multi-agent task execution scenario. In our scenario, tasks can be broken down into different subtasks, each requiring certain expertise or capability to be completed. Meanwhile, each agent can improve its capabilities either by performing the subtask or observing other members solving the subtask in the team. Agents are autonomous. Consequently, each agent tries to improve its chance for getting selected in a task by improving the quality of its capabilities that maybe needed for future task. In this thesis, we propose four task-selection strategies considering potential learning gain differently, and three more task-selection strategies that also consider agent and task openness. We also consider different agent types and different degrees of openness of environment. Agent types are a pre-defined set of agents including novice agents, average agents, and expert agents. An expert agent is one that has more expert capabilities than an average agent, and so does an average agent over a novice agent. Here we report on our experiments showing the impact of agent and task openness on the environment, agent's

learning and task performance, investigating the roles of the different task selection strategies, and demonstrating the importance and need to consider openness in multiagent ad hoc team formation problems.

3.2 Related Work

In Stone, Gan, et al. (2010), teacher and learner agents try to optimize a team goal (collect maximum amount of cans) where the problem is formulated as an instance of k-armed bandits problem. The teacher agent has to decide on either optimizing its own utility (collect higher number of cans itself), or going for a sub-optimal option in order to teach the learner agent. One key factor is that the teacher and learner agents are always present in the environment and do not leave. If, on the other hand, the agents can leave and new agents can enter the environment, there could be very different implications. Based on how frequently agents leave (or new agents enter) the environment, teaching might have to be done more frequently, less frequently, or even none at all. For example, if an agent is only in the environment for a very short time, then it could be better for the teacher agent to not teach, and instead improve its own utility as it does not make sense for the teacher agent to teach, when the learner agent might leave quickly, without staying long enough to implement and improve the team's utility with what it has learned. It stands to reason that teaching frequently would be more beneficial only if the learning agent remains in the environment for a longer period of time, actually reaping the benefits of the new knowledge it has gained.

Also, since the tasks in the environment are fixed, there is a guarantee on available tasks, and there are benefits of learning. Our consideration is that of an open

environment, where task openness is considered, e.g., a task might have to be done frequently, or it could be a one-time task only. If the probability of certain task appearing in the system is more frequent, teaching other agents to solve those tasks would be beneficial. If not, then teaching would not be necessary and the knowledge gained to solve that particular task would likely not be used. This means, the decision to teach or not teach, would benefit from taking this factor into account, thus calling for the analysis on task and agent openness.

In Barrett et al. (2011), the research is on how ad hoc agents can perform, especially in the pursuit domain, where the agents are predators, trying to capture a prey. The actions that the agents perform in this environment are to capture the prey. There is an element of learning in the scenario, but this is limited to just on that action of capturing the prey. But ad hoc agents might be required to perform multitude of tasks, requiring different types of skills, thereby making it beneficial for them to learn multiple skills. This consideration of learning multiple skills is not made in Barrett et al. (2011) as those agents do not perform multiple type of tasks, but only a single type of task.

Also, the teams in the scenario described in Barrett et al. (2011) are "static", i.e. agents do not leave or enter the environment. The question we want to answer is, what might happen if agents can come and go as they please? For example, if a predator is replaced by a new predator, it would require other teammates to learn about the new predator teammate. Indeed, this dynamism in the environment motivates our research towards analyzing how the performance of teams is affected by the introduction of open environment in terms of tasks and teammates.

3.3 Simulation Framework

3.3.1 Multiagent System Design

We model our ad hoc environment using three main components. A set of existing tasks (tasks inside of the environment), $\mathcal{T}$, a set of existing agents (agents inside of the environment), A, and a blackboard-based publish-subscribe system. In our environment, existing agents can communicate and collaborate through the blackboard without knowing each other beforehand. For example, to form a team to solve a task $T \in \mathcal{T}$, agents need to bid for T in an auction held by the administrator of the environment on the blackboard, without direct communications with other agents. Therefore, agents bidding for T have no idea of with whom they will work until after the auction results are disclosed. Agents who win the auction may, consequently, work with other agents they have never met before. Agents can also access current tasks information through the blackboard to assist their decision-making. The environment is managed by an administrator (admin). New agents are introduced into the environment and some existing agents are removed from the environment, based on the Agent Openness (AO) parameter, by the admin. The admin obtains new agents from an agent's pool outside of the environment. Removed agents from the environment will not be sent back to the agents' pool. New tasks are also introduced into the environment, based on the Task Openness (TO) parameter, by the admin obtaining or sampling new tasks from a tasks pool.

3.3.2 Openness

Task Openness (TO). Task openness affects the relative values of immediate versus delayed task rewards and outcomes. Many aspects of real world collaborative group processes involve a time delay between when a decision is made and when the benefits for that decision are realized. Decisions may require balancing of short-term gain or success versus future potential gains or successes. In the environment, agents making decisions in team formation have to tradeoff between current and future task rewards, as less-than-optimal rewards for a current task may produce higher rewards for other tasks in the future. For example, in a resource-constrained environment, an agent might withhold its resources from optimally solving a current task *T1*, with the expectation that it would use the resources to solve a future task *T2* that has a higher reward. However, if the environment has high task openness, trading off current rewards for future ones might not be a good idea, as *T2* might never appear in the environment again. These inter-temporal choices are inherent in virtually all decision-making contexts, and inclusion of the openness of the tasks in the environment is therefore critical for effective modeling of team functioning. In the real world, task repetition and scheduling, as well as the occurrence and evolution of new tasks and requirements, are some of the various reasons that could affect TO.

Agent Openness (AO). Agent openness affects decisions of team members to collaborate or to share their knowledge or expertise or learn from others. Teams may involve members with different types and skillsets, often diverse in their makeup in real-world situations. As such, agents may have heterogeneous sensing, reasoning, and acting

capabilities that may or may not be known to the other agents in the first place. In such situations, multiagent learning approaches have been proposed for agents to learn from each other and even to share knowledge (e.g., teach) with each other (Barrett et al., 2011). However, deciding to learn or teach is not trivial. Let us consider two cases. Case 1: Suppose *A1* has to decide whether to join one of two teams, *C1* or *C2*. Joining *C1* would give a higher reward; however, joining *C2* would give *A1* an opportunity to work with and learn from a high-capability agent *A2*. Case 2: Suppose that *A1* has to decide whether to share knowledge with another agent *A3*, with the idea that if *A1* shares knowledge with agent *A3* now, the benefits from working with an improved *A3* in future teams would outgain the expense. Such considerations are certainly valid and could lead to optimization of rewards. However, what if *A2* is not capable of sharing in Case 1, and *A3* exits from the environment in Case 2? Then *A1*'s decision to join *C2* (Case 1) would be unwise and its efforts to teach (Case 2) would be all for naught. Thus, modeling such AO of the environment can help knowledge sharing and can help optimize the learning. In real-world situations, equipment faults, sensor downtimes, instrument malfunctions, personnel changes, and role re-assignments are some of the factors impacting AO.

Simulating Openness. In our experiments, we simulate both AO and TO by introducing new agents and tasks in our simulation. We randomly remove agents from the simulation and introduce agents that were not previously present in the simulation in order to implement AO. The rate at which we remove the agents in the simulation and introduce newer agents depends on AO, $AO \in [0,1]$, where 0 means no new agent is introduced and 1 means the all the N_a agents that exist at the time $t = 0$ will be replaced by the end of the simulation with different agents. In general, the number of agents

removed at each clock tick is $\lfloor (N_a/T') * \mathrm{AO} \rfloor$ where T' is total simulation ticks. (Note that $(N_a/T') * AO$ is not always an integer, in which case we accumulate the floored decimal values, when it reaches 1, then we remove one more agent from the environment at that tick.) TO is also simulated by introducing tasks which have different sub-tasks and difficulty as the simulation moves forward, $TO \in [0,1]$. One new task is added to the system at each tick in the simulation and TO = 0 means that each new task has already appeared before in the environment and TO = 1 means each new task is a different task from the ones already in the environment (i.e., tasks which have different combinations of subtasks and difficulty).

Agents Perceiving Openness. For the purposes of our experiments, to investigate the impact of considering openness when an agent makes decisions, we use the ideal assumptions that agents know exactly the actual values of AO and TO. Note that this is not necessarily true in real world ad hoc situations, and we will address this as future work. In our design, the admin publishes the AO and TO on blackboard so that every agent receives the "ground truth". We term this approach "informed perception".

3.3.3 Tasks and Capabilities

We define $\mathcal{T}$ be a set of all tasks in the environment, each task $T \in \mathcal{T}$ is determined by the subtasks comprising the task. Let $\overline{\tau}$ denote the set of all subtasks in our environment, so we have $T = \{\tau_1, \tau_2, \cdots, \tau_{|T|}\}$, where $\tau \in \overline{\tau}$. Similarly, $\mathcal{C} = \{c_1, c_2, \ldots, c_{|C|}\}$ denote the set of all capabilities that agents could have. Each subtask $\tau \in \overline{\tau}$ requires exactly one capability c from the set $\mathcal{C}$ to solve. For example, in order to solve subtask τ_k, the capability c_k is needed. Also, in our design, each subtask τ_k

is associated with two more parameters, the minimum number of agents n_k that are required to perform the subtask τ_k, the minimum quality threshold $qt_k \in (0,1]$ that agents are required to have in order to perform the subtask. A subtask τ_k can appear in many different tasks, with possibly different n_k and qt_k. Furthermore a set of agent is denoted as A, and each agent $a_i \in A$ is described by $\boldsymbol{cap_i} = \langle cap_{i,1}, cap_{i,2}, \cdots, cap_{i,|\mathcal{C}|} \rangle \in [0,1]^{|\mathcal{C}|}$ where $cap_{i,k}$ denotes a_i's expertise with respect to the k-th capability c_k.

3.3.4 Learning

Learning happens in several ways. In our simulation, we focus on two types. We assume that it is inevitable that an agent (e.g., a human) would learn some of the experiences and expertise of certain skills either from practicing (learn by doing) or from watching its collaborators performing the tasks (learn by observation). This learning process is likely to lead to changes in an agent's capabilities and subsequent decision making. To this end, we adopt the following learning framework based on principles from human learning theory (Khandaker & Soh, 2007). When a person practices some skills, when he/she does not have much expertise in the beginning, the room for improvement is relatively large and the learning gain is apparent. As they gain more experience and become better and better, the improvement becomes harder and the learning gain also diminishes. Following this theory, we designed our learning by doing, using Equation. 3.1, with agent a_i on capability k.

$$Gain_{self}(a_i, k) = \frac{\eta}{cap_{i,k}+\varepsilon} \qquad (3.1)$$

where η is a constant denoting the increment in knowledge from self-learning and ε is a small number in case $cap_{i,k}$= 0. This gives the amount of capability increase of agent a_i on capability k.

Moreover, in human learning scenarios, when a person learns from another, the amount of information transferred between two agents is proportional to the knowledge gradient between them (Jumadinova et al., 2014). Following this approach, we model the learning gain by a learner agent, a_l, from interacting with a practicing agent, a_t, on capability k to be proportional to the capability difference between them, $cap_{t,k} - cap_{l,k}$. *Note that as we do not consider explicit teaching in the current simulation, we do not identify practicing agents as "teacher" agents.*

Designing an appropriate function to quantify the learning gain while modeling human learning requires some insight. Vygotsky's zone of proximal development (ZPD) theory (Vygotsky, 1978) suggests that it may be difficult for two persons to teach/learn from each other if the amount of prior knowledge they have on a topic is vastly different from each other or almost identical to each other. At the same time, as the learner's knowledge increases, the amount of learning gain that it can obtain also diminishes, as its knowledge starts to converge with that of the teacher. Based on this theory, we design the learning gain function of agent a_l observing agent a_t successfully completing a subtask k as in Equation 3.2 below.

$$Gain_{Observe}(a_l, a_t, k)$$

$$= \begin{cases} 0 & if\ x < 0 \\ -\frac{\beta}{\alpha^2}x^2 + 2\frac{\beta}{\alpha}x & if\ 0 \le x < \alpha \\ -\frac{\beta}{(\alpha-1)^2}x^2 + \frac{2\alpha\beta}{(\alpha-1)^2}x + \frac{\beta(1-2\alpha)}{(\alpha-1)^2} & if\ \alpha \le x < 1 \end{cases} \quad (3.2)$$

where x is the capability difference between agent a_t and agent a_l, $x = cap_{t,k} - cap_{l,k}$ and β is the maximum learning gain that a_l can acquire from observing agent a_t, and α is the capability difference that gives the maximum learning gain (when $x = \alpha$, the learning gain is β, which is the maximum learning gain). With the function described in Equtation. 3.2, we can see that when the capability difference x is small (between 0 and α) the learning gain drops rapidly as x gets smaller from α to 0 and the learning gain reaches 0 when the two agents have equal knowledge.

Finally, we define the total learning gain of an agent a_i, when working in a team, as in Equation 3.3.

$$Gain(a_i) = \sum_{\tau_m \in T} Gain_{Self}(a_i, \tau_m) + \sum_{\tau_n \in T \setminus \{\tau_m\}} Gain_{Observe}(a_i, a_{jmax}, \tau_n) \quad (3.3)$$

where we assume $a_{jmax} \in A_T \setminus \{a_i\}$, where A_T denotes all the agents that are assigned to solving task T, $j_{max} = \arg\max_j Gain(a_i, a_j, k)$ for a particular capability k. This means that if a_i observes more than one agent completing a subtask τ_n, we will use the agent a_{jmax} to determine most learning by observation gain for a_i.

Note that a key difference between the above learning by observation approach and learning by being taught as modeled in Stone, Gan, et al. (2010) is that when an agent considers potential learning gain here, the agent implicitly tries to put itself in a situation where it would be more likely to learn from observing others to improve its capabilities and agents in our design presently do not have to worry about whether to teach, whereas an agent in Stone, Gan, et al. (2010) has to reason explicitly about teaching. Moreover, our agent design only considers how to improve an agent's own capabilities and not others as in Stone, Gan, et al. (2010). Nevertheless, teaching in Stone, Gan, et al. (2010) does not require specific contract or agreement from the agents to be taught, and thus parallels our learning by observation at least in spirit. And in our future work we will integrate agent teaching of Stone, Gan, et al. (2010) to more completely capture learning occurring in ad hoc teams.

3.3.5 Task Selection Strategies

In our simulation design, tasks are allocated through auctions held on blackboard. Agents can see the available tasks as well as tasks' specification. Then, based only on this information and agents' perception of AO and TO, agents make decisions on which task to bid on. When an agent chooses a task to bid, it needs to consider several things: (1) the direct task rewards for helping completing the task, (2) the learning rewards/gains it can get both from practicing its skills when executing the subtask (learning by doing) and from observing its team members completing other subtasks (learning by observation), and (3) the uncertainties in the environment, as captured in the environmental openness (AO and TO)—more specifically, the expected availability of agents from whom the

capabilities can be learned via working in a team and the type of tasks that would likely appear in the future. To this end, we propose the following task selection strategies. These strategies are based on the assumption that the system administrator—i.e., auctioneer—assigns each subtask $\tau \in T$ to the agents who bid on the task T with the best matching capability.

In the following, the first three task selection strategies are based on the evaluation of the subtasks' quality requirements and the agents' quality of corresponding capabilities only; there is consideration for neither AO nor TO. The next four strategies do consider environmental openness. In the following, let T_{best} denote the task that an agent chooses to bid on that is to its best interest. Each agent a_i has a vector, $\boldsymbol{cap_i} = \langle cap_{i,1}, cap_{i,2}, \cdots, cap_{i,|\mathcal{C}|} \rangle$, and $cap_{i,k}$, denotes the kth capability of agent a_i in $\boldsymbol{cap_i}$.

Strategy 1. Most Qualified (MQ)

$$T_{\text{best}} = \arg\max_{T} \left(\sum_k cap_{i,k} - qt_k \right) \qquad (3.4)$$

Notice here, we sum over k, where $k \in \{k | cap_{i,k} > qt_k \; and \; \tau_k \in T\}$. With this MQ strategy, we find the total of positive differences of agent a_i's corresponding capabilities of subtasks and the quality requirement of subtasks in each task T. Since an agent a_i is capable of doing a subtask τ_k, then this agent must have its $cap_{i,k} - qt_k > 0$, and the bigger the difference is, the more qualified it is for this subtask.

Strategy 2. Most Learning Opportunity (MLO)

$$T_{best} = \arg\max_{T}(U_{oberve}(T)) \tag{3.5}$$

$$U_{oberve}(T) = \frac{\sum_{k'}\left|cap_{i,k'} - qt_{k'}\right|}{n_o} \tag{3.6}$$

where $k' \in \{k' | cap_{i,k'} < qt_{k'}, \tau_{k'} \in T\}$, $U_{oberve}(T)$ is the potential utility that the bidding agent can gain from observing other teammates executing the subtasks, and n_o is the number of subtasks observed. Note that not all $T \in T_{available}$ are candidates for an agent to apply this strategy. If agent a_i does not have a subtask τ_k s.t. $cap_{i,k} - qt_k \geq 0$, then a_i will not consider bidding for this task.

Strategy 3. Most Qualified + Learning (MQ+LO)

This strategy is a hybrid of the first two strategies. Agents not only consider the opportunity to learn from other agents by observation but also consider their qualification for solving one subtask within a task.

$$T_{best} = \arg\max_{T}\left(U_{learn}(T)\right) \tag{3.7}$$

where $U_{learn}(T) = (U_{doing}(T) + U_{observe})/2$, $U_{doing}(T) = \left(cap_{i,j_{max}} - qt_{j_{max}}\right)$, $j_{\max} = \arg\max_{j}\left(cap_{i,j} - qt_j\right)$, $\tau_j \in T$ and $cap_{i,j} - qt_j \geq 0$. Note that $U_{doing}(T)$ is the expected utility of the bidding agent for executing its best qualified subtask of T. It computes the largest positive difference of agent a_i's $cap_{i,j}$ and the required quality threshold qt_j. This term considers this agent's qualification of its best quality that matches the task's required capabilities. $U_{oberve}(T)$ is same as defined in Eq. 3.6. Similar to *MLO*, if there is no $j_{\max}$ for task T, then the agent does not bid for it.

Before introducing the next set of task selection strategies, here we define a key term called the total potential utility of participating in the solution of a task T in Equation 3.8.

$$U(T) = w_L \cdot U_{learn}(T) + w_S \cdot U_{solve}(T) \qquad (3.8)$$

where w_L and w_S are the weights for learning and solving a task, respectively, and $w_L + w_S = 1$. $U_{learn}(T)$ is the potential utility from learning by doing and learning by observation as defined in Eq. 3.7 above. $U_{solve}(T)$ is the potential utility of the bidding agent participating in solving the task T, as in Eq. 3.9:

$$U_{solve}(T) = \rho \frac{qt_{jmax}}{\sum_k qt_{k*n_k}} \cdot R_T \qquad (3.9)$$

where $j_{max} = \arg\max_j \left(cap_{i,j} - qt_j\right), \tau_j \in T$ and $cap_{i,j} \geq qt_j$, and ρ is an adjustment factor to put the $U_{solve}(T)$ in roughly the same range as $U_{learn}(T)$in our simulations. Notice in the denominator, we sum the required quality threshold of each subtask, $\tau_k \in T$, multiplied by each agent number requirement n_k, to model the difficulty level of a task. R_T is a parameter that represents the reward for completing the task T. In the case that there is no j_{max} then $U_{solve}(T) = 0$, and the agent does not bid for this task since it is not qualified for it.

The following task selection strategies are all based on the total potential utility. Notice in Equation. 3.8, there are two parameters w_L and w_S, which are the weights for learning and solving a task, respectively. An agent's perception of the environmental openness, AO and TO, could and should affect its decisions on task selection through shifting the weights w_L and w_S. For example, in the case that an agent perceives that *AO*

is high—which means agents come and leave very frequently, the likelihood for the agent to, say, work with the same agent again to learn a particular capability is low. So, in such a scenario, an agent might want to learn things as much or as soon as possible so that they can acquire the capability to solve the task that are highly likely to appear again in the future to gain more utilities. Therefore, it is prudent to increase the weight of w_L to emphasize more on learning. On the other hand, if the agent perceives that *TO* is high, then the tasks change very frequently. In such environment, agents do not have to learn to solve a particular task—say, a difficult one—if the task would not likely appear again in the future, then the likelihood of encountering new tasks which require different capabilities could be very high. In that case, the agents might not care too much about learning to solving particular tasks, and aim for getting more rewards sooner. Therefore, shifting more weight to w_L to focus on getting immediate rewards makes more sense.

Strategy 4. Most Total Potential Utility (MTPU)

$$T_{best} = \arg\max_{T} \big(U(T)\big) \qquad (3.10)$$

where $U(T)$ is the total potential utility as defined in Eq. 3.8. Within this *MTPU* strategy, we have several interesting variants by setting the weights differently: ***Strategy 4.1. MTPU_L=S*** with $w_L = w_S = 0.5$; ***Strategy 4.2. MTPU_L<S*** with $w_L = 0.25, w_S = 0.75$, and ***Strategy 4.3. MTPU_L>S*** with $w_L = 0.75, w_S = 0.25$;

Strategy 5. MTPU with Agent Openness (MTPU+AO)

This strategy is also based on Eqtation. 3.9, but taking *AO* into account. As we mentioned above, when agents come and go frequently (*AO* is high), putting more

attention on learning certain capabilities from certain agent before it leaves the environment might be a wise decision. Hence, for the MTPU+AO strategy, we set $w_L = AO$ and $w_S = 1 - AO$.

Strategy 6. MTPU with Task Openness (MTPU+TO)

Similarly, using the same Equation. 3.10, but taking TO into account. When TO is high, focusing on immediate rewards is a good choice. Hence for this strategy, we set $w_L = 1 - TO$ and $w_S = TO$.

Strategy 7. MTPU with Both Openness (MTPU+ATO)

Similarly, using the same Equation. 3.10 for the MTPU+ATO strategy, we use $w_L = \frac{AO}{AO+TO}$ and $w_S = \frac{TO}{AO+TO}$. We define w_L and w_S in Strategy 7 as such so that when AO and TO are either high or both low, the weight for learning (w_L) and the weight for getting the immediate rewards (w_S) are not too different from each other.

On the other hand, when AO is high and TO is low, we will get $w_L > w_s$. In this case, agent-leaving rate is high. The chance for encountering a particular agent to learn a particular capability is slim. Hence seizing the opportunity to learn some particular capability before its gone might be critical. Meanwhile, the tasks (task types) more or less will be the same over time, due to the low TO. Hence, learning a particular skill or capability, say a useful one that is currently in demand, is promising to bring more future benefits. These two considerations both suggest that focusing more on learning might be a better choice.

Conversely, when *TO* is high and *AO* is low, it will result in $w_s > w_L$. In this case, tasks are changing rapidly and different tasks usually require different skills to perform. So, it might not make much sense for an agent to learn some particular capabilities, since those capabilities might not be needed again. It might be then wiser to focus on getting more things done and getting more rewards now. In addition, agents are more stable in this case due to low *AO*; there is more chance to learn some particular skills from some agents, since the agents who have special capabilities tend to stay around longer in the environment. Hence there is no need to worry about the "expertise" to solve a particular subtask of a task to disappear from the environment. These two reasons suggest that agent solving the task and getting immediate rewards might deserve more attention.

3.4 Results

3.4.1 Configuration Parameters

Before we can analyze the roles of task openness and agent openness in the ad hoc teams, we need to come up with two important configuration parameters for our simulations: (1) the distribution of different agent types—expert, average, and novice—in the system and (2) a configuration of required time for agents to finish a task (t_T) and the required number of agents to finish a task (n_T) to facilitate the feasibility of completing high number of tasks and achieving sufficiently high total learning gain. Since these parameters are set to afford us meaningful, comparable results for a wide range of openness levels, we dub this configuration Facilitator Configuration (FC). For this effort, we used task selection strategy 3 (MQ+LO) and defined an expert agent as one with at

least one of its initial capabilities in the range of [0.7 to 1.0], an average agent of range [0.3 to 0.7], and a novice agent in the range of [0.0 to 0.3].

After running experiments with different mix of expert, novice and average agents, we realized that a uniform distribution of the agent types—33.33% for each type— would result in better task completion and total learning than with configuration having higher number of expert or average agents. The uniform agent configuration would mean that there are balanced numbers of expert agents to help complete tasks and of average and novice agents involved in completing tasks and learning.

Next, we used the above uniform agent configuration to run simulations with different values of t_T and n_T. The simulation results are provided in Tables 3.1 and 3.2. From Table 3.1, we see that the scenario with t_T= 25 and n_T = 1X produced highest learning efficiency for all agent types. From Table 3.2, the same configuration also performed well in terms of tasks completion with 74.53% task completion rate, with highest task completion rate for novice agents and very high task completion rate for average agents. Lower n_T allowed agents to form teams which were capable of solving most tasks. 25 ticks to finish a task might look counter intuitive as shorter tasks are easier to be completed. But, if the tasks are really short then only expert agents would be involved in solving them, thereby decreasing the learning gain, as novice and average agents would not have opportunities to win any task whatsoever owing to the availability of expert agents all the time. Consequently, the configuration with t_T = 25 and n_T = 1X was chosen for our experiments. Finally, in addition to using the Facilitator Configuration, we also used the following parameters in our experiments: *AO* = (0, 0.25, 0.5, 0.75, 1.0), *TO* = (0, 0.25, 0.5, 0.75, 1.0), number of agents per simulation run is 900,

one task is introduced per time tick, and the number of non-zero initial capabilities for each agent = (1, 3, 5). This last parameter models an agent's ability to solve tasks when it is first created.

Table 3.1 Simulation results in terms of total learning gain achieved to determine Facilitator Configuration. 1X means the number of agents required to complete each subtask of a task is 1, 2, or 3; 2X means it is 2, 4, or 6; and so forth

n_T		t_T Number of Ticks (in 10^{-2})				
		1	**10**	**15**	**20**	**25**
1X	**N**	0.000	3.698	4.331	4.900	6.152
	A	2.337	3.151	3.612	3.973	4.039
	E	1.038	2.377	2.694	2.878	2.909
2X	**N**	2.294	2.954	3.553	3.419	3.012
	A	1.641	2.469	2.470	2.513	2.470
	E	1.250	1.887	1.869	1.894	1.933
3X	**N**	2.177	2.315	1.230	2.971	2.627
	A	1.677	2.004	1.803	1.926	2.027
	E	1.275	1.368	1.463	1.456	1.424
4X	**N**	1.901	2.079	1.675	1.898	1.427
	A	1.534	1.429	1.434	1.570	1.802
	E	1.206	1.074	1.198	1.209	1.161

Table 3.2 Simulation results in terms of number of tasks solved to determine Facilitator Configuration. 1X means the number of agents required to complete each subtask of a task is 1, 2, or 3; 2X means it is 2, 4, or 6; and so forth

n_T		t_T Number of Ticks (in 10^{-2})				
		1	**10**	**15**	**20**	**25**
1X	**N**	0.00	3.30	8.33	10.43	7.57
	A	0.97	68.73	98.97	114.2	110.17
	E	1027.6	952.1	873.97	758.7	656.67
2X	**N**	2.80	13.80	8.27	6.70	2.97
	A	51.13	156.1	127.50	100.5	100.20
	E	1893.8	1023	783.50	600.5	540.43
3X	**N**	6.77	3.80	2.00	3.17	3.37
	A	173.33	110.0	85.57	80.60	71.70
	E	1715.2	693.7	501.93	426.8	361.03
4X	**N**	6.67	1.63	1.47	2.03	1.6
	A	150.9	52.2	51.23	43.93	46.13
	E	821.5	361	325.6	264.8	228.53

3.4.2 Experiments and Results

Here we report on three experiments. The first experiment was designed to investigate the roles of agent openness (AO) and task openness (TO) in task completion and learning. More specifically, we wanted to study how agents can change their team forming decisions when we increase *AO* only, *TO* only and both *AO* and *TO*. Also, we wanted to investigate the compounding effects of combining different levels of *AO* and *TO* with respect to task completion and learning gain. For this experiment, we used the number of non-zero initial capability = 5 as it made agents more capable of solving the tasks but not too easily, as well as enabled average and novice agents to contribute in task solving. The second experiment was aimed to gain insights into how the different task selection strategies, as described in Section 3.3.5, would perform under different combinations of *AO* and *TO* by studying their performances in task completion and learning gain. The third experiment was designed to investigate how changing the number of non-zero initial capabilities would change the overall performance of the agents and the roles of AO and TO.

For the above three experiments, we used the Facilitator Configuration described in Section 3.4.1, we set the total number of ticks per simulation to 500 to enable agents to make use of their learned capabilities in task solving. Also, we set $|\mathcal{C}| = 20, |T| = 5$ for all $\in \mathcal{T}, \rho = 5,\ \eta = 0.01, \varepsilon = 0.001, \beta = 0.05, a = 0.25$ and $R_T = 1$ for our equations outlined in Section 3.3.4. There were 25 AO and TO combinations (Section 3.3), 9 task selection strategies in total, 3 values for non-zero initial capabilities options (1, 3, and 5) and we ran 30 times for each AO-TO-task selection strategy combination. This yielded a total of 20,250 simulations (25×9×3×30).

3.4.2.1 *Investigating Roles of AO and TO in Task Completion and Learning*

Figure 3.1 illustrates the roles of openness in ad hoc team setting. First, as the tasks in the environment became more open—i.e. new tasks requiring different skills emerging in the environment, both task completion rate and learning gain decreased. There was a pronounced decrease in the total number of task completed (T) and learning gain (L) along the y-axis. This reinforces the hypothesis that in an ad hoc scenario, it is crucial to consider TO. We also see that the task completion rate decreased by 63%, 77%, 80%, and 85% when TO increased from 0.0 to 1.0 with a step of 0.25, for AO = 0. Correspondingly, learning gain also decreased by 70%, 82%, 85%, and 89% at the same time. This decrease could be attributed to the fact that when TO increased, the agents needed to solve newer problems, requiring skills which might not yet be available among them. This led to tasks not being auctioned off, decreasing the tasks completion as well as opportunities for learning. We see an analogy of the observation in a disaster response scenario, for example. Suppose there are doctors, engineers, and firemen in the volunteer team, but the situation demands them to navigate through a minefield, this can severely limit the tasks that the team can complete. This can lead the team to abandon certain region beyond the minefield, thereby decreasing the tasks completion rate and learning gain as they do not have the necessary skills to complete those very specific jobs.

Also from Figure 3.1, we can observe that Task Completion (T) and Learning Gain (L) numbers generally increased when AO increased, but only when TO > 0. Learning gain actually decreased and task completion remained constant when AO increased if *TO* = 0. Both of these trends go on to show that the environment in which ad hoc teams operate could have a more complex impact on an agent's reasoning or decision making

than how they are currently being considered. When $TO > 0$, new tasks are introduced to the environment, which, on average will require some new skills to be completed. This is where increasing AO is beneficial, as newer agents, on average, will bring some new skills to the environment, which might be relevant in solving the newer tasks. This is still so even though, on average, the expertise/skills lost from expert agents leaving and replaced with new average or novice agents tended to average out the expertise gained from new expert agents entering the environment and average or novice agents leaving the environment. Consequently, when $TO > 0$, both task completion rate and learning gain generally increased with when AO increased. On the other hand, when $TO = 0$, no new type of tasks was introduced to the environment. So, increasing AO would result in higher net loss of capabilities on average as new agents came in to solve the same old problems, whereas older agents that would have solved tasks, and as a result learned some capabilities, would leave the environment. This behavior was somewhat unexpected, and went on to show the complexity of considering environmental openness in ad hoc scenario. This is why we observe that introducing new agents—i.e., increasing AO did not necessarily help learning if newer tasks were not being introduced to "motivate" the agents. Hence learning gain (L) decreased when AO increased in Region III, but task completion rate was not affected as there was enough expertise in the environment to solve all the tasks (500 in total).

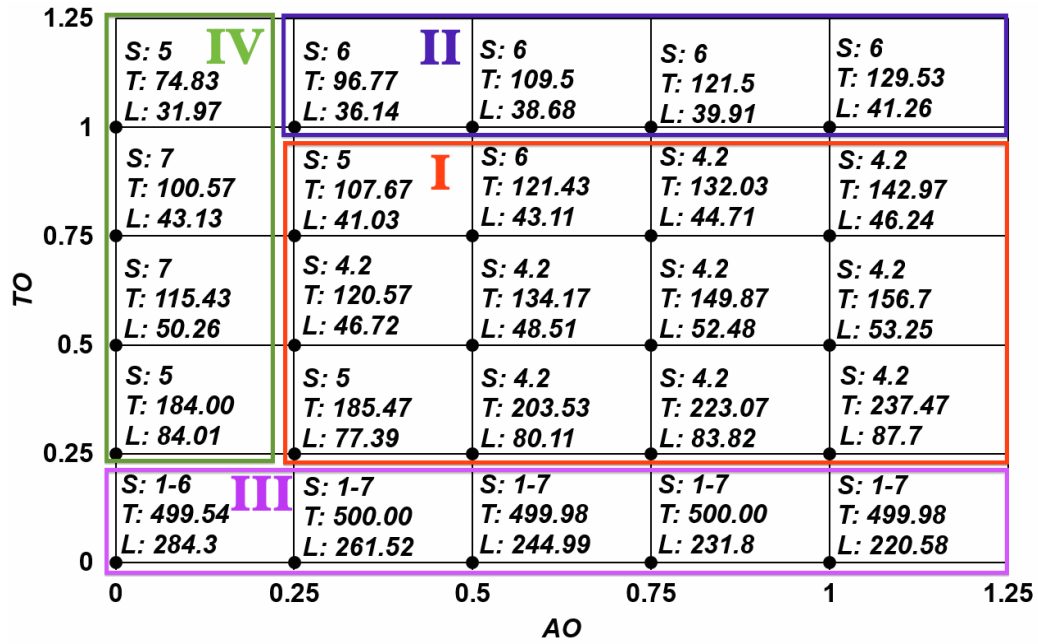

Figure 3.1 Task selection strategy with best task completion and learning gain per *AO*-*TO* combination with the number of non-zero initial capabilities = 5. *S* = best performing task selection strategy, *T* = # of total tasks solved, *L* = total learning gain.

3.4.2.2 *Investigating Task Selection Strategies' Performance in Environment with Different AO and TO*

As per Figure 3.1, it is clear that no one task selection strategy dominates all situations. For different combinations of *AO* and *TO*, the best task selection strategy varies. In general, we can divide our observations of the results in Figure 1 into four regions: Region I: $0.25 \leq AO \leq 1$ and $0.25 \leq TO \leq 0.75$; Region II: $0.25 \leq AO \leq 1$ and $TO = 1$; Region III: $0 \leq AO \leq 1$ and $TO = 0$; Region IV: $AO = 0$ and $0.25 \leq TO \leq 1$. First, in Region I, when the tasks in the environment became more open but not yet completely open (i.e., $TO = 1$), and the agents in the environment became more open, we observed that the strategy *MTPU_L<S*—note that other strategies (*MTPU+AO* and *MTPU+TO*) shown in Region I all were reduced to the same Strategy *MTPU_L<S*—

which weighs potential learning utility ($w_L = 0.25$) smaller than the potential tasks solving utility ($w_S = 0.75$), performed the best. This result shows that as both agents and tasks in the environment are open, the learning utility plays a less important role than tasks solving utility in terms of tasks completion. When *AO* is non-zero, new agents are introduced and old agents leave with their learned capabilities. As a result, agents have fewer opportunities to use their new learned capabilities to solve tasks before they leave. Combined with the fact that new, previously unknown tasks were also introduced into the environment, the learned capabilities may not be used for these new introduced tasks. Thus, *MTPU_L<S*, by emphasizing task solving more than learning, was able to perform better than other strategies.

Second, in Region II, when tasks in the environment were completely open, the observed best strategy is *MTPU+TO*, which takes only TO into the account when estimating total potential utility. Since $TO = 1$ in this region, the weight of learning utility became 0, as a result, this strategy actually only considered the tasks solving utility. Upon further consideration, this result was actually expected as new tasks were always different with previously seen tasks when $TO = 1$, the capabilities agents learned from previously seen tasks were more likely to be not applicable for the new, unknown tasks. Hence considering learning did not necessarily benefit the potential rewards in the future and the strategy *MTPU+TO*, which would maximize the immediate reward in this situation, turned out to be the best performing one.

Third, in Region III, we observed that the best strategy varied. Actually, our simulation data shows that there were no best strategies for any *AO-TO* combinations in this region. In this region, though not shown in the graph, all strategies indeed performed

equally well, except for Strategy 7 (*MTPU+ATO*). The difference between the best performing strategies and the worst performing strategies in terms of task completion was within 0.001% and the difference in terms learning was within 0.01% for all *AO-TO* combinations in this region for Strategies 1-6. However, for Strategy 7, recall that at $AO = 0, TO = 0$, the weights for considering task completion and learning would be 0 and thus Strategy 7 ended up with agents not bidding for any task. Note also that when the tasks were closed, i.e., $TO = 0$, and no new tasks were introduced into the environment, agents were solving the same types of tasks all the time; hence every task selection strategy produced very similar results for every task.

Fourth, in Region IV, where $AO = 0$ and $TO > 0$, the best strategies were *MTPU+AO* (Strategy 5) and *MTPU+ATO* (Strategy 7). Notice that in this region, $AO = 0$, which made both strategies simplify to consider only the utilities from solving the tasks. In this region, agents were closed, i.e., no agents would leave and no new agents would enter the environment. Hence agents did not have to worry about agents with expertise from whom it could learn useful skills becoming unavailable. Indeed, the expertise would always stay in the environment as reliable resources in such a situation. Thus, agents focusing getting more immediate rewards would be able to leverage that— as in Strategies 5 and 7— to their advantage, as observed in this region of Figure 1. Therefore, when $AO = 0$ and $TO > 0$, the task selecting strategies which put emphasis on the utilities from solving the tasks such as *MTPU+AO* and *MTPU+ATO* were the best choices.

Based on these observations, we see that agents considering AO and TO in their task selection strategies could indeed improve their utilities, that these agents could

leverage the dynamics in the environment to their advantage in ad hoc team formation. However, a key issue not addressed in our design is that right now, we used the "informed perception" of AO and TO. What would happen if agents were required to perceive both openness on their own? How would they bootstrap their task selection when they did not have sufficient data to model both openness? Or would agents give up on using AO and TO in their strategies if they realized they could not perceive them accurately due to incomplete information? We aim to investigate different ways of perceiving openness in our future work.

3.4.2.3 *Investigating the Impact of Number of Nonzero Initial Capabilities*

How would the agents perform differently if they were created with different number of nonzero initial capabilities? That is, if they were more capable or less capable at the start of the simulation, would different task selection strategies perform differently and would the impacts of AO and TO be mitigated or magnified?

For this experiment, we refer to Figure 3.1 where the number of nonzero initial capabilities was 5—i.e., with more capable agents—and Figure 3.2 where it was 1—i.e., with less capable agents. Note that we also ran a set of simulations using 3 nonzero initial capabilities per agent, but observed that its results were essentially the same as those shown in Figure 3.1.

Comparing Figures 3.1 and 3.2, we see two key differences. First, when tasks were closed, i.e., $TO = 0$, along the x-axis, as agents became more open, there was a consistent increase in T observed in Figure 3.2 but not in Figure 3.1. The reason for this increase is because agents were solving the same type of tasks, and due to the fact that the

environment lacked expertise—as agents in Figure 3.2 were less capable, new agents coming into the environment could bring in capabilities that were not present in the environment. This would help the agents solve some previously unsolvable tasks.

Second, in Region I, while steady trends were observed in Figure 3.1, it is not so in Figure 3.2. For example, we see that both *S* and *L* increased as AO increased, and both decreased as TO increased, in Figure 3.1. But in Figure 3.2, such trends were not apparent. We speculate that because of low-capability agents in Figure 3.2, due to the lack of opportunities to qualify for and thus solve tasks, the agents also did not have sufficient opportunities to learn. And thus, this also implies that considering AO and TO in tasks selection strategies might not be worthwhile.

Third, where $TO = 0.25$, and $AO = (0.25, 0.5, 0.75, 1)$, *MTPU_L<S* (Strategy 4.2)—note that Strategy 5 reduced to Strategy 4.2 at $AO = 0.25$—performed best in Figure 3.1 whereas *MLO* (Strategy 2) did so in Figure 3.2. Upon further consideration, we realize that when the number of non-zero initial capabilities = 1, as *TO* was sufficiently low (e.g., 0.25), agents tried to learn because there was a relatively higher chance of seeing old tasks in the environment, and improving skills that were required for those tasks, which in turn would improve their chance of actually solving the tasks. Thus, agents selecting tasks emphasizing learning performed better as in Strategy 2 (*MLO*). On the other hand, when the number of non-zero initial capabilities = 5, even when *TO* was very low, agents still had a relatively higher variety of skills, enabling them to concentrate on solving tasks rather than learning. This is reflected by *MTPU_L<S* (Strategy 4.2), which focuses more on task solving (75%) than on learning (25%).

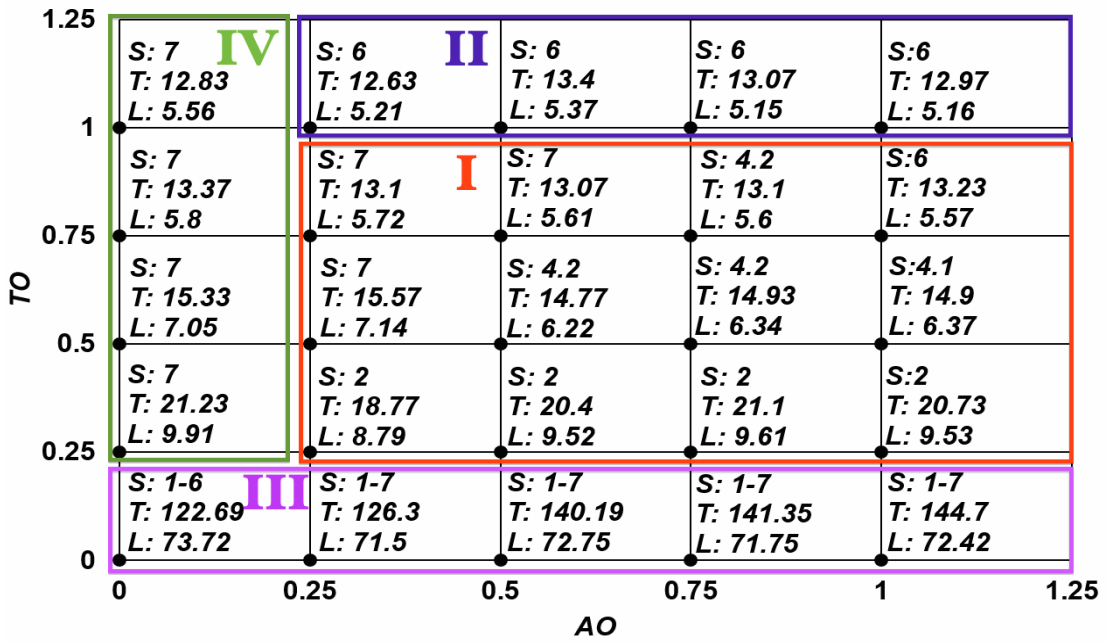

Figure 3.2 Task selection strategy with best task completion and Learning gain per AO-TO combination with number of non-zero initial capabilities = 1. S = best performing task selection strategy, T = # of total tasks solved, L = total learning gain.

3.5 Conclusions

We have developed an ad hoc team formation framework that takes into account learning and task solving under varying degrees of environmental openness. The learning involved is based on "learning by observation" and "learning by doing" modeling learning theory on the zone of proximal distance. An additional emphasis here is about how an agent can choose a subtask to do such that joining a team to help complete an overall task allows the agent to position itself to gain from learning, from doing the subtask and from observing others working in the team. Furthermore, we have devised mechanisms to simulate agent and task openness. Running simulations of this framework, we were able to study various effects of considering agent openness (AO) and task openness (TO) in ad-hoc team formation. We were able to see that AO and TO are

important in ad hoc team formation. Based on how the task completion rate as well as learning gain varied with different levels of AO and TO, it is clear that these two factors should be considered to more comprehensively represent real world ad hoc teams. First, AO and TO change the way teams are formed. With environment being open, agents need to factor in the possibility of new agents and tasks entering the environment in order to make better decisions in terms of joining a team. Second, AO impacts learning, with the introduction of new agents specially boosting the learning when new tasks are also being introduced into the environment. TO makes it difficult for agents to solve the tasks. The possibility of new tasks emerging in the environment means newer agents entering the environment can be helpful as they could bring newer capabilities.

Having now established the importance of AO and TO, gaining insights into the relationship between the two factors, and investigating the effectiveness of several openness-based task selection strategies, we have identified several key next steps to continue with this line of research. First, we will explore more realistic ways to perceive openness—as our "informed perception" scenario where agents know both AO and TO exactly, is not ideal—such as (1) NoSharing, where agents model on their own without sharing information with each other, (2) Sharing, where agents share information to model the openness together. Second, as reported in Section 3.4.2.1, we see for $TO = 0$, increasing *AO* decreases the learning gain. Furthermore, as reported in Section 3.4.2.3, we see no clear effect of increasing *AO* in Regions I and II from Figure 2 We need to further explore this emergent behavior to better understand the complex relationship between AO and TO in ad hoc teams. Third, we will consider the impact of both teaching and learning while modeling agent's behavior, particularly incorporating the fundamental

game-theoretic work from (Stone, Gan, et al., 2010). This will require agents to consider the potential gain from teaching another agent, as opposed to only considering potential gain from learning from other. Fourth, we will consider agent reliability in terms of agent possibly failing to complete tasks to incorporate (perceived) solution robustness into agent reasoning when bidding for tasks, with little or no knowledge of the capabilities of other agents in the ad hoc team formation environment.

Chapter 4: Collaborative Human Task Assignment for Open Systems

4.1 Introduction

Intelligent agents and multiagent systems have been used in a wide variety of application to support human activities and decision making. For instance, there are autonomous personal assistants that support their users in carrying out tasks, managing schedules, and so forth. For example, Chalupsky et al. (2002) and Tambe et al. (2008) described Electric Elves that helped humans in accomplishing organizational activities, such as rescheduling meetings, selecting presenters for research meetings, tracking people's locations, and organizing lunch meetings. Myers et al. (2007) described a system that relieved the user of routine tasks and intervened in situations where cognitive overload leads to oversights or mistakes by the user. Berry et al. (2006) described a personalized agent called PTIME for time management and meeting scheduling as part of a larger assistive agent system called CALO. There are also collaboration support systems aimed at identifying for human users other human users to help with problem solving, teamwork, and learning. For example, Vassileva et al. (2015) described PHelpS that helped workers find appropriate helpers among their peers when they were encountering problems while interacting with their database, and I-Help that matched students with their peer helpers for university courses. Khandaker et al. (2011) described computer-supported collaborative learning applications called I-MINDS and ClassroomWiki to form optimal student teams based on students' tracked and modeled behaviors. Finally, Sklar and Richard (2006) pointed out, in addition to peer learning agents, that there were also pedagogical agents and demonstrating agents used in human learning systems. *Pedagogical agents* (Heidig & Clarebout, 2011) are designed to

facilitate learner motivation and learning. They act as tutors and model student learner profiles and the current state of knowledge to customize their interactions accordingly.

One particular problem that agents are well suited to assist human users with is **collaborative task assignment**, where there exist a set of human users and a set of tasks that require multiple people to combine their individual skills and expertise to work together towards a common, temporary goal, earning each participant a share of a joint reward if the task is accomplished successfully. In such a problem, a multiagent solution is advantageous because agents representing individual human users can first model the abilities of their assigned users, then find and acquire tasks that best benefit their users, while at the same time fairly allocate tasks across all users so that the overall system also benefits. For example, agent-based human collaborative task assignment could be used to (1) form temporary teams of freelance workers (e.g., independent software developers or artists) to satisfy contracts from companies lacking the internal expertise to accomplish tasks (e.g., developing particular pieces of software or graphic design), (2) combine the expertise and skills of office workers across divisions within large companies to accomplish tasks needed by the company, or (3) further improve matching students to peer-based learning tasks in computer-aided education.

However, collaborative task assignment becomes much more challenging within dynamic, open environments where the system itself changes due to entities coming and going over time. In particular, we consider two types of openness affecting the collaborative task assignment problem. First, **agent openness** occurs whenever the set of human agents changes as people join and leave the environment over time. This causes expertise and skills needed to accomplish tasks to become more or less prevalent,

affecting the ability of software agents to find suitable people to accomplish each task. For instance, if an expert and skilled person leaves the environment, then tasks that could be successfully accomplished in the past might not be possible anymore. Second, **task openness** occurs whenever the set of collaborative tasks changes: both new tasks requiring different expertise and skills appear and older tasks disappear over time. People specializing in certain types of tasks might need to adapt what they work on if those tasks disappear, while other people who had difficulty contributing might become more useful as new tasks related to their expertise and skills appear.

Both types of openness cause uncertainty within the collaborative task assignment problem, as software agents do not know which tasks might be successfully accomplished now or in the future due to fluctuations in both the set of people needed to complete tasks, as well as the set of tasks itself. Given that there might be multiple tasks each person could contribute to at any point in time, yet a person can only contribute to one task at a time, openness makes the problem of selecting appropriate tasks for human users more difficult for software agents.

In order to address this difficulty, we propose a solution integrating two important factors into agent reasoning within an auction protocol used to fairly assign people to collaborative tasks. First, software agents model the **uncertainty in task accomplishment** caused by agent and task openness. In particular, agents learn probabilistic models of the likelihood that both (1) its bid will be accepted and thus its person matched to a particular task, and (2) enough people with appropriate expertise and skills will be available so that the task is successfully accomplished. If either the agent's bid fails, or enough people cannot be found to satisfy a particular task, then the agent's

user will not complete a task, reducing the total reward earned by the user. Over a sequence of available tasks, the agent then uses this probabilistic model to bid on tasks that will maximize the users' expected rewards over time.

Second, to further improve reward maximization over time in spite of environment openness, we are inspired by the fact that the expertise and skills of human users are not static, but can improve over time through **human learning** when they complete tasks and interact during teamwork. In particular, we incorporate realistic models combining two types of human learning: (1) *learning by doing* (Henderson, 1984; Leibowitz, et al., 2010; Shell et al., 2010; Ying, 1967) where people gain ability through experience accomplishing tasks, and (2*) learning by observation* (Bandura, 1986, 2004), where people gain ability by watching collaborators perform activities within the same task that are currently too difficult for the user. Such models are factored into the agent's decision about how to bid on tasks, helping each agent choose tasks that will allow its user to improve so that it earns greater future rewards. In short, we see that *factoring in human learning is especially important in open environments, as learning is necessary to counter the possible loss of expertise and skills within the system caused by agent openness, as well as to develop abilities to complete a wider range of tasks introduced through task openness*.

Using a series of experiments, our empirical results demonstrate: (1) the negative effects on collaborative task assignment caused by both agent and task openness, necessitating a solution for handling these challenging properties of real-world environments, (2) the benefits of reasoning about uncertainty caused by openness when finding and selecting tasks for human users to complete, including greater task

accomplishment, and (3) the improvements in cumulative rewards earned by users caused by modeling human learning to promote the non-myopic maximization of task rewards over uncertain, future tasks.

4.2 Collaborative Task Assignment Problem

One application of intelligent agents to assist human users is in collaborative task assignment, where software agents are responsible for finding and acquiring tasks for their human users to complete in collaborative teams. In this section, we describe (1) how we model the collaborative task problem, and (2) how we refine this model to account for the complexities caused by environment openness common to real-world collaborative team assignment problems. In Section 4.3, we will describe how agents model human learning so that they can reason about the improvements in their users' expertise and skills over time.

4.2.1 Problem Model

Environments of the collaborative task assignment problem contain three main components: (1) a set of tasks $\mathcal{T}$, (2) a set of human users H that must collaborate to complete tasks, and (3) a set of software agents A, where each agent $a_h \in A$ is assigned to a unique human user $h \in H$ and is responsible for acquiring tasks for the human h to complete.

We define $\mathcal{T}$ be a set of all tasks in the environment. Each task $T \in \mathcal{T}$ is composed of multiple subtasks that must be completed by human users. We use τ to denote a subtask, and represent each task as a set of subtasks: $T = \{\tau_1, \tau_2, \cdots, \tau_{|T|}\}$.

Let $\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$ denote the set of all capabilities or unique skills that human users could have. For example, in a freelance software developer environment, there could be capabilities defining users' (i.e., freelance programmers') abilities to program in the Java programming language, to write documentation, to design systems, etc. In contrast, in an office setting, there could be capabilities defining users' abilities to send emails, to schedule meetings, to order supplies, etc.

Each subtask τ requires exactly one capability c from the set $\mathcal{C}$ to complete the subtask. Each subtask also has a quality threshold $qt \in (0,1]$ defining how much expertise in the corresponding capability c is needed to complete the subtask. Finally, each subtask also has a number $n \in \mathbb{N}$ defining the number of people needed to complete the subtask. Combining this definition of a subtask, we represent each subtask τ_k as the triple $\langle c_k, qt_k, n_k \rangle$. For notational convenience, we override the notation of a subtask τ_k to denote that it requires $c_k \in \mathcal{C}$.

Furthermore, we define H to be a set of human users in the environment. Each human user $h \in H$ is described by a vector $\boldsymbol{cap_h} = \langle cap_{h,1}, cap_{h,2}, \cdots, cap_{h,|\mathcal{C}|} \rangle \in [0,1]^{|\mathcal{C}|}$ specifying her expertise with respect to each capability, where $cap_{h,k}$, denotes the expertise of human h with respect to the k-th capability c_k. Over time, each human user can improve her capabilities through learning, causing users to be capable of completing a greater number of possible subtasks over time, and thus act as better teammates for collaborative tasks. We describe human learning in more detail in Section 4.3 below.

Within the collaborative task problem, we add a constraint to model the restriction of many real-world environments that a human user can only commit to working on a single collaborative task at a time. This enables the human to focus all of her energies on a single objective at once. Thus, each user's agent does not over-commit its user to multiple tasks and thus avoid having to deal with multiple teams, which could possibly risk task failure due to overly busy users, thereby benefitting the user's collaborative team. However, users can be assigned to multiple subtasks within the same task, if she has the appropriate expertise in $\boldsymbol{cap_h}$.

Human users are motivated to complete as many tasks as possible, as each task T provides a reward $R(h,T) \in \mathbb{R}$ to the human user. Rewards are only earned if the task is successfully completed, meaning that each subtask $\tau_k \in T$ is completed by the required number of users n_k. For example, in a freelance software developer environment, these rewards could be monetary payments for a collaborative team of developers finishing a software project. Since tasks are collaborative and different users contribute differently to tasks, a task's rewards are shared from a total task reward R_T based on a user's contribution to that task:

$$R(h,T) = \sum_{\tau_k \in T} \frac{\delta(h,\tau_k) \cdot qt_k}{n_k \cdot qt_T} R_T \qquad (4.1)$$

where $\delta(h,\tau_k) = 1$ if user h was assigned to subtask τ_k, else 0, and $\sum_{h \in A} R(h,T) = R_T$. Over time, the tasks acquired by an agent for its human user h form a sequence $(T_h^t)_{t=0}^{\infty}$, where T_h^t denotes the task human h was assigned to at time t (and $T_h^t = \emptyset$ if h is not assigned to a task at time t with $R(h,\emptyset) = 0$). Ultimately, user h desires to maximize her cumulative rewards over the entire sequence of tasks:

$$\max \sum_{t=0}^{\infty} R(h, T_h^t) \qquad (4.2)$$

Thus, the objective of software agent a_h is to find and assign its user h to a sequence of tasks over time that maximizes this objective function.

In order to fairly assign tasks to their human users, software agents compete in a contract net-based (Smith, 1980) auction protocol. In this protocol, a subset of the available tasks $\mathfrak{T} \subseteq \mathcal{T}$ are offered for auction and the descriptions of tasks $T \in \mathfrak{T}$ are communicated to all software agents. As described above, agents are constrained to bidding on only a single task to avoid over-committing their human user to multiple collaborative teams (since users cannot back out of tasks if they win multiple bids, which would otherwise cause such tasks to fail). Thus, each agent must select a single task to bid on for its user. To insure fair assignment of tasks for the benefit of the overall system, the amount each agent a_h bids[1] for a task is the capabilities of its user $\boldsymbol{cap_h}$.

The auctioneer—representing the system and not any particular human user—allocates subtasks to agents and their users in a greedy way. For each task, the auctioneer assigns each subtask $\tau_k \in T$ to the n_k agents that bid on the task T with the highest user capability $cap_{h,k}$. In the case that there are not enough qualifying users for a subtask, then this subtask will not be assigned, thus the entire task will not be auctioned off. In other words, it is possible for an agent to win a bid for its user (matching the user to at least one subtask in the task), yet the task as a whole will not be auctioned off (and thus no collaborative team formed) if there are not enough qualifying users for every subtask of that task.

[1] We assume here that the capabilities of a human user $\boldsymbol{cap_h}$ are known by an agent a_h. Depending on the domain, this knowledge could be acquired by a_h interacting with its human user (e.g., to administer tests) or with other users (e.g., feedback from an expert).

4.2.2 Modeling Environment Openness

As introduced in Section 4.1, many real-world applications of collaborative task allocation occur in complex environments that also contain the challenging property of environment openness. In this thesis, we consider two types of openness: both (1) agent openness and (2) task openness. We now describe how we model these types of openness in task allocation problem.

Agent Openness First, agent openness represents the phenomenon that human users (who are also intelligent, non-artificial agents) join and leave the environment over time. For example, in a freelance software development environment, individual developers might leave software companies to do independent freelance work instead, whereas others might switch from being freelance workers to working solely for a software company. Likewise, in an office worker environment, the company might hire new employees and let others go over time. This definition of agent openness is closely related to the definition used in the intelligent agents and multiagent systems literature for software or hardware agents that join and leave complex environments over time (e.g., Huynh et al., 2006; Jamroga et al., 2013; Pinyol & Sabater-Mir, 2013; Shehory, 2001).

Within our problem model, the set of human users H (and their corresponding software agents A) is non-stationary and changes over time. At any point in time, some users might be removed and others might be added. As a result, these sets are extended to reflect the current available humans (and software agents) at a particular time t as H_t (and A_t). We assume that agents are not aware of which of their peers are around at any point in time, nor that the agents even know how many peers they have.

The primary implication of agent openness is that as a new human user h joins the environment, new expertise $\boldsymbol{cap_h}$ becomes available to assist with completing collaborative tasks. However, as an existing human user h leaves the environment, so too does their expertise $\boldsymbol{cap_h}$, potentially making it more difficult for collaborative tasks to be completed. This is especially problematic since human users are capable of learning to improve their expertise over time, so the amount of overall expertise leaving the system due to openness could exceed the amount of expertise joining the system.

Task Openness Second, task openness represents the phenomenon that the set of tasks that require collaboration to solve could also change. For example, in a freelance software development environment, changes in programming paradigms and the types of software needed by clients would cause different collaborative tasks to exist over time. Moreover, in an office worker environment, different seasonal activities of the company could require different tasks over time.

Within our problem model, the overall set of tasks $\mathcal{T}$ is non-stationary and changes over time. At any point in time, some tasks might be removed and others might be added. As a result, this set is extended to reflect the current possible tasks at a particular time t as $\mathcal{T}_t$.

The primary implication of task openness is that as the set of tasks changes over time, different expertise and capabilities are required. As easier tasks become available or difficult tasks disappear, more users will be qualified to complete tasks, whereas when more difficult tasks become available or easy tasks disappear, then fewer users will be qualified to complete tasks. Each of these phenomena affects the ability of agents to

acquire tasks for their users: the former creates more competition between agents for tasks, whereas the latter makes it more difficult to find a suitable task for a user.

Overall, both task openness and agent openness make it very difficult for agents to select tasks for their users that maximize long-term rewards as they introduce uncertainty into both (1) whether the agent will win a bid for a task, which is vital since agents are constrained to a single bid per auction, and (2) what tasks will be available in the future, and thus what types of capabilities its users will need to learn to complete those tasks.

Of note, our work on agent and task openness within a problem model such as that described in Section 4.2.1 is similar to and builds upon prior research by Jumadinova et al. (2014). In particular, their research explored the impacts of agent and task openness when agents work together in ad hoc teams (similar to collaborative human task assignment) under the assumption of simple rules for forming teams based on agent capabilities. Our research, on the other hand, proposes a solution for directly reasoning about the uncertainties caused by agent and task openness, then maximizes the rewards received from collaborative tasks. We also add principled computational models of human learning based on an extensive literature review to improve how agents reason about the benefits of task accomplishment for human users.

4.3 Human Learning Model

To model human learning, we focus on two particular learning paradigms: learning by doing and learning by observation.

Learning By Doing. Learning by doing can be viewed from two perspectives. From an economic theory viewpoint, it is the process of performing a task or carrying out

an action, and learning from that before performing the same task again. It is considered an adaptive approach to multi-period decision making (Ying, 1967). From a cognitive learning viewpoint, it can be seen as repetition, as outlined in the Unified Learning Model (ULM) (Shell et al., 2010), where it is a process by which knowledge is reinforced through repeated access, exposure, or application. Newell and Rosenbloom (1993) stated that "almost always, practice brings improvement, and more practice brings more improvement."

To model learning by doing in our problem, we borrow clues from *experience curve effects* (Henderson, 1984) to derive the learning gain function for a human user performing learning by doing, and *learning curve* to characterize different types of tasks. The experience curve effects indicate that over time, the more units of a good that a company produces, the average cost per unit is lowered, as the people with the company accumulate experience and expertise to better produce such good. Meanwhile, depending on the skills or knowledge that are required to perform or master a task, there are different learning curves. For example, learning how to perform some skills might be quick at first, but difficult to master (e.g., playing strategic games such as Go or chess), whereas others skills might have slow learning at first, then faster with more experience (e.g., learning to ride a bicycle or swim). In short, we see that different task types may impose different learning curves, such as power law, linear, exponential, and sigmoidal (Leibowitz et al., 2010; Newell & Rosenbloom, 1993). In our problem, we use the exponential learning equation for success-based learning outlined by Leibowitz et al. (2010). According to Leibowitz *et al*., a learning equation can be modeled as:

$$p_n = p_\infty - (p_\infty - p) \cdot e^{-\alpha \cdot S_n} \qquad (4.3)$$

where p is the performance measure, n is akin to n-th learning episode, such that p_∞ is the maximum infinite-horizon performance measure achievable, p_0 is the initial performance measure, S_n is the accumulated sum of all previous performances until, but not including, the nth episode, and α is a constant rate coefficient. Mapping these to our problem: p_0 refers to a user h's initial expertise for a particular capability, $cap_{h,k}$; p_n is the current expertise of h after n-times performing that capability; and $S_n = \sum_{i=0}^{n-1} p_i$. The change in the performance measure, or learning gain, according to Leibowitz et al. (2010), is:

$$\dot{p} = \alpha p \cdot (p_\infty - p) \qquad (4.4)$$

The constant rate coefficient α caps the amount of learning gain at each episode. (Note that we will use α_{do} to indicate the rate is associated with learning-by-doing.) The general shape of this curve is a (concave downward) parabola: when a user's expertise is low, it learns a little; as its expertise grows, it starts to learn more with a higher learning gain; then after it peaks, it starts to learn less as its maximum expertise is reached. For simpler tasks, the initial gain is higher (or more steep); and for more complex tasks, the initial gain is lower (Roediger & Smith, 2012; Wifall et al., 2014). Thus, for a user h's gain via learning by doing for performing a subtask with a learning curve capped by α_{do}, using its capability $cap_{h,k}$, we have:

$$\Delta_{do} cap_{h,k} = \dot{cap}_{h,k} = \alpha_{do} \cdot cap_{h,k} \cdot \left(1 - cap_{h,k}\right) \qquad (4.5)$$

In summary, a user's gain in learning by doing is determined by its current capability, the learning curve of the capability being learned, and the total amount of learning depends on the number of times that it has performed the capability.

Learning By Observation. Bandura (2004) described observational learning (or learning by observation) as knowledge acquisition by learning from the examples provided by others. Bandura's social cognitive learning theory (Bandura, 1986) indicated that there are four stages involved in observational learning: attention, retention or memory, initiation or reproduction, and motivation.

In our problem, we model learning by observation in the following manner. A user can learn from observing other users only when they are in the same team collaboratively solving a task. This allows us to model a user's *attention*. To ensure *retention* (or memory), each user updates its capability after task execution. Most importantly, to model *initiation* (or reproduction), "observers must be physically and intellectually capable of producing the act." That is, even when an observer user receives the stimuli from its observation of the performing user, reproducing the observed action may involve skills that the user does not yet have. Thus, we model the learning gain function of user h observing a teammate j performing subtask τ_l as follows:

$$\Delta_{obs} cap_{h,l} = \begin{cases} 0 & otherwise \\ \dot{p} & 0 \leq qt_l - cap_{h,l} < \beta \end{cases} \quad (4.6)$$

where β is the threshold under which $qt_l - cap_{h,l}$ is small enough for learning by observation to take place, and $\dot{p}$ for observational learning is modeled similarly from Eqs. 4.3-4.4 above:

$$\dot{p} = \alpha_{obs} \cdot \left(qt_l - cap_{h,l}\right) \cdot \left(\beta - \left(qt_l - cap_{h,l}\right)\right) \quad (4.7)$$

where α_{obs} refers to the cap for the corresponding learning curve for observational learning for that capability. Note that it is possible for a capability to have different

values of α_{do} and α_{obs} as a capability could be easier when it is learned by doing than when it is learned by observation and vice versa. In summary, gain from learning by observation is zero if a user observes a subtask being performed that requires a much higher level of capability ($\geq \beta$). Also, if a user is already capable of performing the subtask, then it does not learn anymore from observing another user performing the subtask. Further, a user's learning gain from observational learning follows the same sigmoidal curve as for learning by doing, albeit stunted by β.

4.4 Solution

Given the above descriptions of both the collaborative task assignment problem and a mathematical approach for modeling human learning within collaborative tasks, we describe our solution for agent-based reasoning to acquire tasks for human users.

4.4.1 Estimating Expected Task Rewards

Recall that in the collaborative task assignment problem, an agent a_h's objective is to maximize the cumulative reward (Eq. 4.2) earned by its user h over the sequence of tasks acquired by a_h through bidding in the task auction. This requires non-myopic planning.

However, due to uncertainty caused by agent openness, estimating the reward a user would earn from a particular task $R(h, T)$ if the user were assigned to the task and it were completed is difficult because the agent does not know which other users exist in the environment and thus what bids their agents would make and who would be assigned to different subtasks $\tau_k \in T$.

Instead, the agent needs to estimate an *expected* task reward that accounts for this uncertainty. We can model this as:

$$E[R(h,T)] = \sum_{\tau_k \in T} \frac{P_h(\tau_k) \cdot qt_k}{n_k \cdot qt_k} R_T \qquad (4.8)$$

where $P_h(\tau_k)$ represents the probability that user h is assigned to subtask τ_k (assuming that the user is assigned to task T). Unfortunately, this probability is neither directly measurable nor computable due to agent openness.

However, we can rely on the following intuition to address this issue. Given the procedure followed by the auctioneer (c.f., Section 4.2.1), we know that the users with the highest capability $cap_{h,k}$ are going to be assigned to subtask τ_k. Hence, the higher a user h's capability $cap_{h,k}$, the more likely it is to be selected to perform subtask τ_k. Thus, we know that

$$P_h(\tau_k) \propto diff(h,\tau_k) = \max\{0, cap_{h,k} - qt_k\} \qquad (4.9)$$

where $diff(h,\tau_k)$ represents how much more expertise h possesses than required by τ_k. Therefore, we know that maximizing

$$E\left[\hat{R}(h,T)\right] = \sum_{\tau_k \in T} \frac{diff(h,\tau_k) \cdot qt_k}{n_k \cdot qt_k} R_T \qquad (4.10)$$

also maximizes Eq. 4.8. So, we use Eq. 4.10 to estimate expected task rewards $E[R(h,T)]$.

4.4.2 Approximating Future Task Rewards

Maximizing a user's cumulative task rewards (Eq. 4.2) requires not only acquiring the task that maximizes the user's *current reward* when bidding on tasks, but also

maximizing *future rewards*. Unfortunately, estimating future rewards for the human user $\sum_{t=1}^{\infty} R\ (h, T_h^t)$ is even more challenging due to task openness: the agent does not know what tasks will be available in the future. At the same time, the agent needs to consider future rewards when deciding how to bid on current tasks because completing a task now enables the agent's human user to learn (both by doing and by observation) to improve her abilities to complete future tasks.

Although learning thus couples future rewards to current decisions—making planning more challenging as a result of task openness—our solution instead *leverages* this property to approximate future task rewards.

Similar to our intuition in Section 4.4.1 to address expected task rewards, we note that better learning now by a human user will lead to additional opportunities to complete tasks in the future as the user becomes more and more qualified to complete a wider range of possible future tasks. Thus, tasks provide a **total utility** to users that consists of two parts: (1) rewards for completing the task, and (2) expertise gain in user capabilities that will lead to future rewards. From this perspective, we can model the total utility of a task T for a user h as:

$$U(h,T) = R(h,T) + U_{Learn}(h,T) \qquad (4.11)$$

Given the computational model for human learning provided in Section 4.3 (defined in the literature on human learning), an agent models the utility of expertise gain in its human user from a task:

$$U_{Learn}(h,T) = \frac{1}{2}[U_{L-Do}(h,T) + U_{L-Obs}(h,T)] \qquad (4.12)$$

which balances learning by doing subtask τ_k and learning by observing other subtasks $\tau_l \in T$ based on Eqs. 4.5-4.6.

Once again, due to uncertainty in the environment caused by agent openness, an agent will not know which subtask(s) its user will be responsible for if she is assigned to a task, so the agent needs to compute *expected* learning and total utilities:

$$E[U(h,T)] = E\left[\hat{R}(h,T)\right] + E[U_{Learn}(h,T)] \tag{4.13}$$

$$E[U_{Learn}(h,T)] = \frac{1}{2}E[U_{L-Do}(h,T)] + \frac{1}{2}E[U_{L-Obs}(h,T)] \tag{4.14}$$

$$E[U_{L-Do}(h,T)] = \sum_{\tau_k \in T} diff(h,\tau_k)\Delta_{do} cap_{h,k} \tag{4.15}$$

$$E[U_{L-Obs}(h,T)] = \sum_{\tau_l \in T}(1 - diff(h,\tau_k))\Delta_{obs} cap_{h,l} \tag{4.16}$$

where $diff(h,\tau_k)$ again approximates the probability that user h will be assigned to subtask τ_k (and $1 - diff(h,\tau_k)$ approximates the probability that the user is not assigned to subtask τ_k). That is, since $P_h(\tau_k) \propto diff(h,\tau_k)$, maximizing Eq. 4.13 maximizes the expectation of total utility Eq. 4.11.

Putting all of this together, even though an agent cannot estimate which tasks will be available for its user in the future due to task openness, selecting tasks that maximize Eq. 4.13 will balance maximizing both current expected task rewards, as well as the user's learning so that she can accomplish more tasks in the future. As a result, the agent reasons non-myopically as desired and approximately optimizes the user's cumulative reward function Eq. 4.2 (where exact optimization is impossible due to agent and task openness).

4.4.3 Estimating Uncertain Task Assignment

Thus far, we have developed a solution that enables agents to estimate the expected cumulative rewards over a sequence of tasks (from a current task) for its human user, assuming that the user is assigned those tasks and they are successfully completed. The last step of our solution is to account for uncertainty in task assignment itself, as well as the uncertainty that an assigned task will be successfully completed.

In particular, agent openness also causes uncertainty in whether a user will be assigned to a task if an agent bids on that task because the agent does not necessarily know what other agents with which it is competing to acquire collaborative tasks for its user (where other, more qualified users could instead be amongst the n_k users selected for each subtask τ_k). Moreover, assuming that the agent can win a bid for a task, the agent still does not know whether enough peer users will be found to work with the user on that task in order to have the task successfully auctioned off. Finally, the agent does not know if its user's uncertain peers will successfully complete an assigned task.

To address these uncertainties, our solution models the probability that the agent will acquire a successful task $T \in \mathfrak{T}$ for its user in the current round of bidding as follows. First, we split the probability into three parts: (1) the probability that the agent will win a submitted bid $P_{wb}(T)$ (i.e., the agent is one of the top n_k bidders for some subtask τ_k), (2) the probability that the task will be auctioned off $P_{off}(T|wb)$ (i.e., enough agents with qualified users bid on the task to form a collaborative team), conditioned on the event that the agent wins the bid, and (3) the probability of task success $P_{succ}(T|wb, off)$, conditioned on it being auctioned off to the user.

Using these probabilities, the agent can then compute a refined expected utility for its user from bidding on a task T:

$$E[U(h,T)] = P_{wb}(T) \cdot P_{off}(T|wb) \cdot P_{Succ}(T|wb,off) \cdot \left(E\left[\hat{R}(h,T)\right] + E[U_{Learn}(h,T)]\right) \quad (4.17)$$

To operationalize the probabilities $P_{wb}(T)$, $P_{off}(T|wb)$, and $P_{Succ}(T|wb,off)$, an agent *learns* these probabilities based on its experience in the auction process over time as the environment changes due to both agent openness and user learning affecting the assignment of tasks to suitable users.

To learn $P_{wb}(T)$, the agent considers its recent history from bidding on similar tasks. If the agent won a large number of previous bids on similar tasks, then it has strong evidence that it is one of the most capable agents with respect to this task, and thus it will likely win a bid on task T as well. Likewise, if it lost many previous bids on similar tasks, then the agent should believe it has a low probability of winning a bid on task T. Based on this intuition, the agent considers the s-most similar tasks $S(T)$ that it previously bid on (where task similarity is calculated using the Euclidian distance between the qt_k and n_k values required for the subtasks $\tau_k \in T$). Within these s tasks, it considers the proportion of won bids:

$$P_{wb}(T) = \frac{1}{|S(T)|+\epsilon'_{wb}} \sum_{T' \in S(T)} won(T') + \epsilon_{wb} \quad (4.18)$$

where ϵ_{wb} and ϵ'_{wb} are small constants providing a non-zero (albeit small) probability of winning a bid, even if the agent has never previously won a similar task (noting that its situation might have changed due to human learning and agent openness).

To learn $P_{off}(T|wb)$ we take a very similar approach: counting the number of similar tasks where the agent won the bid and the task was auctioned off (due to enough agents bidding to form a collaborative team with their users):

$$P_{off}(T|wb) = \frac{1}{|S(T)|+\epsilon'_{off}} \sum_{T' \in S(T)} auctionedOff(T') + \epsilon_{off} \qquad (4.19)$$

Finally, to learn $P_{Succ}(T|wb, off)$ we take a very similar approach as well: counting the number of similar successful tasks where the agent won the bid and the task was auctioned off:

$$P_{Succ}(T|wb, off) = \frac{1}{|S(T)|+\epsilon'_{Succ}} \sum_{T' \in S(T)} succeed(T') + \epsilon_{Succ} \qquad (4.20)$$

Overall, Eq. 4.17 (through Eq. 4.18-4.20) accounts for the various different types of uncertainty on collaborative task accomplishment caused by agent and task openness, as well as uncertainty caused by human user learning. Maximizing this function should approximately maximize the human user's cumulative rewards (Eq. 4.2), which is otherwise impossible to optimize directly due to these uncertainties.

4.5 Experimental Setup

To evaluate the performance of our solution in a range of collaborative task assignment problems, we conducted a series of experiments using simulated human users. We compared our approach against baseline agents in order to evaluate the benefits of both (1) our probabilistic modeling of uncertainty caused by agent and task openness within expected utility calculations, and (2) considering the impact of for human learning towards future task accomplishment. In particular, we considered three agent types:

Myopic Baseline (MB): an agent that chooses tasks maximizing Eq. 4.8 without considering human learning or the likelihoods of task success

Learning-Aware Baseline (LAB): an agent that chooses tasks maximizing Eq. 4.13, considering human learning but not the likelihoods of task success

Uncertainty and Learning-Aware (ULA): an agent that chooses tasks maximizing Eq. 4.17, considering human learning and itself learning the likelihoods of task success based on past experience

To evaluate these approaches, we considered three performance measures: (1) the ***number of tasks*** successfully completed per user, evaluating overall system performance, (2) the ***average rewards*** earned per user, evaluating the performance of agents in maximizing their objective function (Eq. 4.2), and (3) the average ***learning gain*** per user, evaluating the ability of agents to choose tasks that also benefit users' future tasks.

To consider the effects of a range of environments with different amounts of agent and task openness, we varied the amount of agent openness (AO) and task openness (TO) present in the environment. $AO \in \{0.0, 0.01, 0.02, 0.05, 0.1\}$ was defined as the proportion of users $h \in H$ who left the environment before each bid, as well as the proportion of agents entering the environment at the same time. Likewise, $TO \in \{0.0, 0.01, 0.02, 0.05, 0.1\}$ was defined as the proportion of tasks $T \in \mathcal{T}$ that disappeared and appeared before each bid. Thus, to aid in evaluation, the number of users and tasks was held constant at $|H| = |\mathcal{T}| = 100$ every round of bidding, even though the contents of these sets changed over time. Each round t, $|\mathfrak{T}| = 20$ tasks were randomly sampled from $\mathcal{T}_t$ and auctioned off to the agents. Each task was composed of 5

subtasks randomly sampled from 20 total capabilities, requiring random $qt \sim [0.1, 1.0]$. Different capabilities had different learning curves for human users, randomly sampled from $\alpha_{do} \sim \{0.1, 0.2, 0.3, 0.4\}$ and $\alpha_{obs} \sim \{1, 2, 3, 4\}$, with $\beta = 0.2$. Total task rewards were randomly sampled within [1, 100]. For each AO, TO, and agent type combination, we conducted 100 experimental runs each with 100 rounds of task bidding. Of note, to focus our evaluation on other aspects of agent reasoning, all assigned tasks succeeded in our experiments.

4.6 Results

We report our empirical results along two perspectives: (1) analysis of the overall system performance with respect to environmental impacts due to agent and task openness and (2) comparison of the three different agent types to investigate the benefits of both reasoning about uncertainty caused by openness in complex environments, as well as considering human learning in the agent's decision making. Note that in the following, we average the performance metrics by the amount of time a user lived in the environment, i.e., a user's lifespan. This is important to provide a fair comparison because of agent openness. For example, it was possible for a user to live in the environment for a very short period of time and thus did not have as many opportunities to participate in the task bidding and completion, gaining rewards and learning.

4.6.1 Impact of Agent and Task Openness

Table 4.1 presents the average numbers of tasks solved per user for different combinations of agent openness and task openness in the environment. The results are further shown for the three different agent types, as outlined in Section 4.5 above. We

see that as AO increased, the average number of tasks solved by users *decreased*. Although AO could cause low expertise users to leave and high expertise users to join at any point in time, our results indicate that the overall trending effect of AO was to cause expertise gained over time through human learning to leave the environment. Thus, the environment lost more expertise than it could recover from incoming human users, limiting the number of tasks that could be successfully completed by collaborative teams.

However, as the amount of TO increased, there were no general, consistent trends. Instead, sometimes the number of tasks completed increased with TO, and other times it decreased. This appears to be evidence that TO has complex interaction effects with AO and agent types.

In summary, we see that openness (especially AO) has adverse impacts on the number of tasks completed by users. In the next section, we will further analyze how different considerations by agents in choosing which tasks to assign to their users affected the benefits to human users.

Table 4.1 Average Number of Tasks Completed Per User With Standard Errors (Normalized by User Lifespan)

Task Openness	Agent Type	Agent Openness 0.0	0.01	0.02	0.05	0.10
0	ULA	0.3052 (0.0005)	0.2906 (0.0004)	0.2830 (0.0004)	0.2695 (0.0004)	0.2597 (0.0004)
	MB	0.2645 (0.0004)	0.2566 (0.0004)	0.2576 (0.0004)	0.2549 (0.0004)	0.2543 (0.0004)
	LAB	0.2606 (0.0004)	0.2544 (0.0004)	0.2543 (0.0004)	0.2529 (0.0004)	0.2490 (0.0004)
0.01	ULA	0.2989 (0.0005)	0.2880 (0.0004)	0.2811 (0.0004)	0.2698 (0.0004)	0.2580 (0.0004)
	MB	0.2665 (0.0004)	0.2586 (0.0004)	0.2514 (0.0004)	0.2509 (0.0004)	0.2513 (0.0004)
	LAB	0.2639 (0.0004)	0.2574 (0.0004)	0.2560 (0.0004)	0.2527 (0.0004)	0.2524 (0.0004)
0.02	ULA	0.2940 (0.0004)	0.2857 (0.0004)	0.2762 (0.0004)	0.2684 (0.0004)	0.2597 (0.0004)
	MB	0.2619 (0.0004)	0.2558 (0.0004)	0.2564 (0.0004)	0.2532 (0.0004)	0.2483 (0.0004)
	LAB	0.2625 (0.0004)	0.2615 (0.0004)	0.2557 (0.0004)	0.2493 (0.0004)	0.2492 (0.0004)
0.05	ULA	0.2869 (0.0004)	0.2797 (0.0004)	0.2735 (0.0004)	0.2637 (0.0004)	0.2576 (0.0004)
	MB	0.2622 (0.0004)	0.2595 (0.0004)	0.2564 (0.0004)	0.2554 (0.0004)	0.2532 (0.0004)
	LAB	0.2634 (0.0004)	0.2596 (0.0004)	0.2562 (0.0004)	0.2525 (0.0004)	0.2532 (0.0004)
0.10	ULA	0.2826 (0.0004)	0.2756 (0.0004)	0.2682 (0.0004)	0.2640 (0.0004)	0.2571 (0.0004)
	MB	0.2621 (0.0004)	0.2578 (0.0004)	0.2578 (0.0004)	0.2563 (0.0004)	0.2544 (0.0004)
	LAB	0.2629 (0.0004)	0.2575 (0.0004)	0.2566 (0.0004)	0.2522 (0.0004)	0.2515 (0.0004)

Table 4.2 Average Reward Per User With Standard Errors (Normalized by User Lifespan)

Task Openness	Agent Type	Agent Openness 0.0	0.01	0.02	0.05	0.10
0.0	**ULA**	2.4374 (0.0045)	2.3717 (0.0045)	2.3101 (0.0045)	2.1917 (0.0044)	2.0887 (0.0043)
	MB	2.1429 (0.0043)	2.1013 (0.0043)	2.0907 (0.0043)	2.0566 (0.0043)	2.0408 (0.0042)
	LAB	2.1090 (0.0043)	2.0833 (0.0043)	2.0895 (0.0043)	2.0335 (0.0042)	2.0088 (0.0042)
0.01	**ULA**	2.3904 (0.0045)	2.3161 (0.0045)	2.2860 (0.0045)	2.1948 (0.0044)	2.0895 (0.0043)
	MB	2.1482 (0.0043)	2.1240 (0.0045)	2.0810 (0.0043)	2.0332 (0.0043)	2.0042 (0.0042)
	LAB	2.1467 (0.0043)	2.0964 (0.0043)	2.0614 (0.0043)	2.0669 (0.0043)	2.0250 (0.0042)
0.02	**ULA**	2.3523 (0.0045)	2.3276 (0.0045)	2.2589 (0.0045)	2.1955 (0.0044)	2.0965 (0.0043)
	MB	2.1288 (0.0043)	2.0857 (0.0043)	2.0763 (0.0043)	2.0527 (0.0043)	2.0166 (0.0042)
	LAB	2.1356 (0.0043)	2.1200 (0.0043)	2.0859 (0.0043)	2.0263 (0.0043)	2.0253 (0.0042)
0.05	**ULA**	2.3326 (0.0045)	2.2690 (0.0045)	2.2268 (0.0044)	2.1474 (0.0044)	2.0781 (0.0043)
	MB	2.1327 (0.0043)	2.0987 (0.0043)	2.0824 (0.0043)	2.0813 (0.0043)	2.0335 (0.0042)
	LAB	2.1488 (0.0043)	2.1101 (0.0043)	2.0850 (0.0043)	2.0440 (0.0043)	2.0480 (0.0042)
0.10	**ULA**	2.3203 (0.0045)	2.2658 (0.0045)	2.2014 (0.0044)	2.1384 (0.0043)	2.0718 (0.0043)
	MB	2.1333 (0.0043)	2.1007 (0.0043)	2.0859 (0.0043)	2.0742 (0.0042)	2.0461 (0.0042)
	LAB	2.1219 (0.0043)	2.0933 (0.0043)	2.0794 (0.0043)	2.0388 (0.0043)	2.0276 (0.0042)

Table 4.3 Average Learning Gain Per User (Normalized by User Lifespan)

Task Openness	Agent Type	Agent Openness 0.0	0.01	0.02	0.05	0.10
0.0	**ULA**	0.00314	0.00359	0.00393	0.00437	0.00458
	MB	0.00279	0.00308	0.00336	0.00390	0.00434
	LAB	0.00279	0.00300	0.00336	0.00389	0.00423
0.01	**ULA**	0.00340	0.00376	0.00403	0.00442	0.00456
	MB	0.00286	0.00319	0.00338	0.00392	0.00430
	LAB	0.00286	0.00318	0.00346	0.00392	0.00434
0.02	**ULA**	0.00351	0.00388	0.00406	0.00440	0.00467
	MB	0.00294	0.00324	0.00354	0.00394	0.00433
	LAB	0.00295	0.00332	0.00350	0.00393	0.00430
0.05	**ULA**	0.00375	0.00407	0.00419	0.00444	0.00461
	MB	0.00308	0.00340	0.00358	0.00410	0.00437
	LAB	0.00312	0.00341	0.00363	0.00399	0.00440
0.10	**ULA**	0.00389	0.00418	0.00425	0.00465	0.00468
	MB	0.00317	0.00345	0.00371	0.00417	0.00446
	LAB	0.00317	0.00344	0.00369	0.00409	0.00437

4.6.2 Comparison of Agent Types

First, with respect to task completion (shown in Table 4.1), modeling the uncertainty in securing successful task assignment by the ULA agents led to significantly greater task completion than the LAB and MB agents. However, as AO increased, the ULA agents' performance still decreased, but maintained higher performance than the other agents. In a way, we see that while the ULA agents were able leverage its modeling of uncertainties in the open environment, such modeling was still to an extent susceptible to the increasing openness in the environment. With increased AO or TO, the strain on maintaining an accurate probabilistic modeling of task success also increased.

Turning more towards the benefits to individual human users represented by software agents, Table 4.2 presents the average rewards received by each user in the environment. First, we see that the ULA agents outperformed the LAB and MB agents with statistical significance. The relatively similar performances between the LAB and MB agents imply that the consideration of expected utilities from learning and solving tasks did not provide marked advantage for the LAB agents over the MB agents that only considered expected utility from solving tasks. The ULA agents' ability to model the uncertainties in the open environment was the difference maker. We will return to this point shortly.

To further compare the three agent types, we look at the amount of learning per user, as shown in Table 4.3. Again, the ULA agents were the most effective among the three agent types, with statistical significance.

4.6.3 Summary

Overall, the ULA agents outperformed the LAB and MB agents in terms of task completed, rewards received, and learning gains. We conclude that software agents probabilistically modeling the uncertainties in open environment is important to achieving better rewards long term for their human users. Although this result is expected, it is especially promising due to the challenges associated with probabilistic modeling in open environments. Recall that due to AO and TO, it is impossible for an agent to directly measure the probabilities in subtask and task assignment. Instead, we had to approximate these through (1) considering the user's capabilities compared to tasks ($diff(h, \tau_k)$) and (2) by learning from the agent's experience bidding in the environment (Eqs. 4.18-4.20). It is very welcoming to observe that these types of clues and learning can be used to help agents model unmeasurable uncertainty in environments with challenging types and amounts of openness.

Interestingly, we also note that as AO or TO increased, all three agent types *increased* their learning gains. Thus, human learning (as modeled from the literature) provides users with a *natural* mechanism to adapt to open environments, acquiring greater quantities of expertise when it is most needed (either due to learned expertise leaving the environment with AO or from more diverse tasks requiring more diverse expertise with TO).

However, referring back to Table 4.1, we see that this increased learning *did not translate into more tasks* solved per user. We believe that this was because as users acquired more expertise and skills, they became qualified for more tasks, leading to more choices for submitting agents' bids. Without explicit coordination, this could have led to

increased competition between agents when trying to acquire tasks for their users (since more users were qualified for the same tasks). In turn, this would result in lost opportunities to work as a team on other tasks on which some competing agents could have instead bid. Thus, while increasing openness facilitated more learning gains in users, it also caused fewer tasks to be solved. We believe that this emergent behavior is rather unique, brought on by openness in the environment, and we will investigate further in our future work. We do note that modeling and learning the probabilities of bid outcomes provided some relief from this problem in ULA agents, since they directly reasoned about the likelihood of being assigned a task (learning from the choices of other agents in prior bidding rounds as a form of implicit coordination).

4.7 Conclusions and Future Work

In this chapter, we have described a multiagent solution for agent-based collaborative human task assignment. We have particularly addressed agent openness and task openness in this problem. We have further modeled human learning by doing and by observation, and incorporated these into the agent's reasoning about how to acquire tasks for its user. Our solution develops an approach for modeling and learning unmeasurable uncertainty caused by environment openness to guide its decision making in maximizing human user reward and learning gains over sequences of tasks. Experimental results demonstrate that our Uncertainty and Learning-Aware (ULA) agents are capable of choosing tasks maximizing expected utilities taking into account the uncertainties and learning. In particular, our ULA agents outperformed two baseline agent types with statistical significance in terms of tasks completed, rewards received, and learning gains.

In terms of future work, first, we will investigate the inflection point of when too much learning is detrimental. We pointed this out in our summary of our results above. Without coordination, more improved human users would bid for more different tasks as they try to maximize their long term utility since they become more qualified for more different tasks. This could cause human expertise to be spread too thin such that only a few tasks can be successfully auctioned off and executed. Perhaps, an agent that supports its human user would need to have metareasoning to decide when to learn and when not to learn based on its success rate of completing tasks. Second, we plan to study the impacts of amount of diversity in the task types and in the agents' capabilities. For example, would considering human learning be able to counter the adverse impacts of openness in the environment if there was only a small percentage of highly capable human users in the environment to begin with? If no, then how many highly capable human users would a system need to be able to "bootstrap" itself up to deal with openness successfully? Diversity of capabilities in the human users can play a role in how the system adapts. Likewise, diversity of task types can affect how human users learn and their ability to complete tasks. Finally, we also plan to further investigate the impact of learning by doing with learning by observation. Learning by observation, in particular, can benefit from more diversity of human users and task types in the environment. Are there alternative models of observational learning? Should software agents specifically model the expected learning utilities of these two types of learning, which our current solution does not do?

Chapter 5: Implementation

In multiagent ad hoc team formation of human like agents, environmental openness plays a crucial role, as agents factorize openness in calculating current versus future rewards to make team formation decisions. In this chapter, we describe a simulator called Multi Agent Ad-Hoc Team Formation Simulator (MAAHTFormS), for implementing and studying various strategies agents can use to make team formation decisions in open environments. We provide a comprehensive description of our simulation environment, and present some of the experiments that can be studied with this comprehensive simulator. MAAHTFormS has been utilized in the research of ad hoc team formations (Chen, B. Chen, X. Timsina, & Soh, 2015; Chen, Eck, & Soh, 2016) and has potentials for aiding further research work in this area.

5.1 Introduction

Ad hoc team formation in multiagent systems has been analyzed with focus on teaching and learning (Stone, Gan, et al., 2010) as well as performance optimization and environmental openness (Jumadinova et al., 2014). As ad hoc team formation is collaboration without pre-coordination (Stone, Kaminka, et al., 2010b), there are many complex factors that needs to be considered in the team formation process (Khandaker & Soh, 2007; Stone, Gan, et al., 2010; Stone, Kaminka, et al., 2010b).

Jumadinova et al. (2014) talk about the need to consider environmental openness for agents to make optimal decisions about team formation. Stone and Kraus (2010) analyze teaching and learning by agents, and if it is better for the system when agents teach or not. There is a whole host of work done on multi agent ad hoc teams, solving

well-established algorithmic problems like k-armed bandits (Stone, Gan, et al., 2010), communication and optimization in pursuit domain (Barrett et al., 2011), and modeling uncertainty in ad hoc team interaction (Agmon et al., 2014).

Since the ad hoc team formation problem in open environment is a complex problem with many factors, there are many levels of experiments, which can be conducted, in order to measure the relationship between those factors (e.g., openness, agent capabilities, agent diversity, and task diversity). MAAHTFormS provides researchers with a comprehensive testing or simulation environment where they can study relationships amongst all the factors impacting ad hoc team formation.

Although some result on the relationship between openness and performance in ad hoc team formation for multiagent system has been studied (Jumadinova et al., 2014), those results are only scratching the surface. There are relationships between environmental openness, learning, and agent behavior within this framework, which needs to be analyzed. Chen et al. (2015) raises one important question about the relationship between agent openness (AO) and task openness (TO), which is yet to be comprehensively studied and understood.

There are also questions regarding agent diversity and task diversity, which are important in ad hoc team formation. Within our framework, agent diversity is the make-up of agents within the simulation, i.e., what percentages of agents have what kind of capabilities (expertise)? Questions regarding the impact of agent diversity on system efficiency and on learning have not been studied. At the same time, tasks within an environment can also be diverse. We believe the diversity of tasks within an ad hoc environment will also affect agent's performance, as diverse tasks need diverse groups of

agents to be completed. The analysis on agent and task diversity will allow us to come up with better agent reasoning within the ad hoc team formation domain.

In the next sections, we review some of the simulators or testbeds for multiagent system found in the literature (Section 5.2), discuss our simulation framework (Section 5.3), detail the implementation (Section 5.4 to Section 5.8) and configuration process (Section 5.9), list the data generated from MAAHTFormS (Section 5.10) and finally include the scripts for running MAAHTFormS on Holland Commuting Center (HCC)'s super computer (Section 5.11).

5.2 Related Work

Jumadinova et al. (2014) introduced a multiagent ad hoc collaboration framework, which considers agent learning in ad hoc environment. This simulation framework allows agent to strategically choose which capability to learn and which agent to learn from. It provided us some insight of considering openness in ad hoc simulation environment. However, this framework has its limitations and needs some more careful treatment in modeling openness. First, this simulation framework introduced two new parameters, "agent openness" and "task openness". but it does not model openness itself. It simply added and removed agents, introduced and replaced old tasks to the environment, instead of reason with openness. Second, agents did not model openness and hence they did not factor the openness into their reasoning when making decisions on choosing tasks. Third, it is a rather simplistic framework to ascertain the impact of openness in the performance of agents in terms of tasks solved and learning. For example, it only considered the total number of subtasks finished and total learning utilities as the impact of openness. On the

other hand, our work also looks at the average values over the number of agents and over the number of ticks, as well as how these averages change over time, and over how agent capabilities change over time. This approach allows us to study the impact of openness in terms of the rate of changes of various metrics. Though the framework considered agent learning, it was based on a function of preset learning utilities and did not consider modeling the effectiveness of learning.

Massaguer et al. (2006) provide a multiagent simulation environment, named DrillSim, in disaster response scenario. It combines drills and simulation into one augmented reality based simulation environment, which evaluates information technology solutions by translating them into disaster metrics (e.g. call delay into time to evacuate etc.). This simulator enables researchers to evaluate many aspects of agent behaviors like cognitive and physical actions, agent's role, etc. In addition, DrillSim provides control over agent role, so that introducing and testing newer agent roles is simpler.

Fullam et al. (2005) describe the various research objectives that must be answered by a testbed for Agent Reputation and Trust (ART) and propose a testbed specification based on those research objectives. They develop a testbed framework that fulfills two purposes (1) comparative study of different research studies on agent trust and reputation, known as competition mode and (2) experiment with single strategies, or utilize result from competition mode for independent study, known as experiment mode. This research provides one important step towards building and using agent testbed for experimentation and evaluation of Agent Reputation and Trust (ART) strategies.

Bouron et al. (1990) propose a testbed to study the interaction between heterogeneous agents. This testbed provides researchers with the ability to control agent types by specifying their architectures and their behavior. There is also the ability to configure different parameters like communication between agents and different types of environment.

The testbeds proposed by Massaguer et al., Fullam et al., Bouron et al. enable different types of experimentation in agent research. The ability to control the types of agent and their communication provided a lot of flexibility in setting up experiments. The simulation framework introduced by Jumadinova et al. (2014) has enabled us to look into agent openness and task openness in the research of ad hoc team formation, but MAAHTFormS provides researcher with the ability to configure, simulate agent team formation experiments, study new environmental factors like agent openness and task openness with agent modeling openness and agent reasoning about openness, and study the effectiveness of learning, as well as studying the impact of other factors like agent and task diversity.

5.3 Simulation Framework

In this section we will discuss the details of the design of our framework, including the design of work flow, agent and task design, the blackboard and auction design. We also include the implementation of simulating of openness, agent perceiving openness, as well as how to configure and run the simulator on Holland Computing Center's (HCC) super computer.

5.3.1 Framework Design

The overall architecture of our system can be found in Figure 5.1. It contains four major components. A set of agents, an admin who controls the environment, and a blackboard-based publish-subscribe system (Blackboard). In our system, agent does not have any knowledge about each other beforehand and does not have any preordinations, they only communicate with each other through Blackboard. Agents inside the simulation environment can see the available tasks' information to assist their decision-making. The overall flow starts by the admin agent introducing agents into the environment from the agents pool and pulling the tasks into the environment from the tasks pool. The task information is published on the blackboard for agents to review. The admin agent controls the environment by introducing new agents and removing old agents according to the agent openness. New tasks are also introduced according to task openness but tasks remain in the environment until they are solved. Agents examine the available tasks and make decisions on which task to bid on according to certain task selection strategies and submit their bids. After that, the auction will start. The admin agent chooses the winning agent according to the algorithm (Algorithm 5.1) with which it is deployed (Section 5.6), and publishes the results to the blackboard for agents to see. After the auction results are disclosed, agents who get selected will gather together (hence form team) to solve the task. The tasks that are not solved (not auctioned off) will remain on the blackboard for the next round of auction together with the newly introduced tasks.

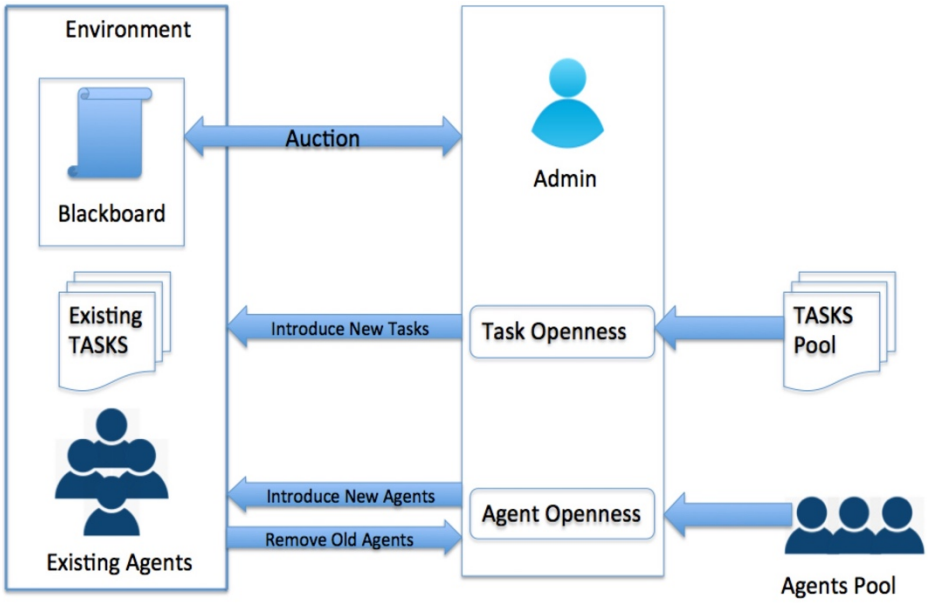

Figure 5.1 Overall architecture of the multiagent simulation system.

5.3.2 Tasks and Capabilities

In Section 3.3.3 we have stated the notations for tasks and capabilities, for the sake of completeness of this chapter, we include them here again in brief.

A set of tasks is denoted as $\mathcal{T}$, and each task $T \in \mathcal{T}$ has a set of subtasks: $T = \{\tau_1, \tau_2, \cdots, \tau_{|T|}\}$. Similarly, $\mathcal{C} = \{c_1, c_2, \ldots, c_{|C|}\}$ denotes the set of all capabilities that agent could have. Each subtask τ requires exactly one capability c from the set $\mathcal{C}$ to complete the subtask. For example, in order to solve subtask τ_k, the capability c_k must be needed. In addition, each subtask τ_k requires $n_k \in \mathbb{N}$ agents to complete and it requires the minimal quality of capability qt_k, where $qt_k \in (0,1]$. Thus, each subtask is a triple $\langle c_k, qt_k, n_k \rangle$. Furthermore a set of agent is denoted as A, and each agent $a_i \in A$ is

described by $\boldsymbol{cap_i} = \langle cap_{i,1}, cap_{i,2}, \cdots, cap_{i,|C|} \rangle \in [0,1]^{|C|}$ where $cap_{i,k}$ denotes a_i's expertise with respect to the k-th capability c_k.

5.3.3 Openness

Openness is the key thing in our study. It represents the phenomenon that agents/tasks join and leave the environment. The focus of this thesis is to investigate the importance and the impact of openness in ad hoc team formation. In this section, we describe the two implementations we used to simulate the openness in Chen et al. (2015) which can be found in Chapter 3, and Chen et al. (2016), which can be found in Chapter 4, respectively.

5.3.3.1 Agent Openness

As stated in Section 3.3.2, agent openness refers to the rate of new, previously unknown agents that are introduced into the environment, while known agents exit the environment.

We have two implementations for agent openness (AO) in our simulator. In both implementations, we randomly remove agents from the simulation and introduce agents that were not previously present in the simulation. The difference is how we remove agents. We use N_a to represent the total number of agents in the simulation environment, and $AO \in [0,1]$, to represent the agent openness. In Chen et al. (2015), $AO = 0$ means no new agent is introduced and $AO = 1$ means all the initial agents introduced in tick 1 will be replaced with different agents by the end of the simulation. Hence, the number of agents to be removed at each tick is $agentPerTick = \lfloor (N_a/T') * AO \rfloor$ where T' is total simulation ticks. Note that $(N_a/T') * AO$ is not always an integer, hence we take the

floored values as the number of agents to be removed, and keep accumulating the decimal values when it reaches 1, then we set $agentPerTick = \lfloor (N_a/T') * AO \rfloor + 1$. This allows us to remove 1 more agent in the current tick.

However, in Chen et al. (2016), we decided to implement the AO as the likelihood of each agent will stay after each tick. At the end of each tick, a uniform random number generator will generate a decimal number between 0 and 1 for each agent. If this number is less than or equal to AO, then this agent will leave and a new agent will enter the environment.

5.3.3.2 **Task Openness**

In Section 3.3.2, the definition of task openness (TO) was given as the rate of new, previously unseen tasks that are introduced into the environment. In Chen et al. (2015), the admin initially posts 30 tasks on the blackboard, then the admin will introduce 1 task at the beginning of each tick. TO determines if this newly introduced task is a new or an old task. A uniform random number between 0 and 1 will be generated and be compared with TO, if the number is less than or equal to TO then the task to be posted should be a new task, otherwise, the task should be an old task. The admin keeps a list of all tasks that are posted, if a new task needs to be posted, then it chooses a task from the task pool, different than the ones that are on the list of posted tasks, to post. If an old task needs to be posted, then the admin randomly chooses a task from the posted task list to post. Notice that, there will be tasks that did not get auctioned off at the end of each tick and they will be re-posted onto the blackboard for the auction for the next tick. In summary, TO is also simulated by introducing tasks which have different sub-tasks and difficulty as

the simulation moves forward, $TO \in [0,1]$. One new task is added to the system at each tick in the simulation and $TO = 0$ means that each new task has already appeared before in the environment and $TO = 1$ means each new task is a different task from the ones already in the environment (i.e., tasks which have different combinations of subtasks and difficulty).

However, in Chen et al. (2016), the admin maintains a task pool of 100 tasks, and it randomly chooses 20 tasks from the task pool to post them on blackboard for auction. Notice, no tasks will be reposted no matter they are auctioned off or not. The TO determines the percentage of the tasks in the task pool (100 tasks) needs to be replaced by the end of each tick.

A third implementation is also included in our simulator. In this implementation, the admin agent posts 20 tasks at the beginning of every tick. The admin agent keeps a list containing the task types of the newly posted 20 tasks. After the tasks have been posted, a uniform random number will be generated for each task on the list, if the number is less than or equal to TO, then this task will be replaced by a brand new task/task type and will be introduced into the environment at the beginning of the next tick. Notice, same as the TO implementation, no matter the task is auctioned off or not, it will not be reposted for the auction.

5.3.4 Agent Perceiving Openness

5.3.4.1 Perceiving Agent Openness

The environmental openness plays an important role on agents considering potential learning gains, and thus indirectly which team to join. In a dynamic

environment, agents or tasks may enter and leave the environment, hence we consider a way to model Agent Openness (AO) and Task Openness (TO) (Section 5.3.3).

Here we provide three options for agents to perceive the environmental openness: (1) *NoSharing*, where agents model on their own without sharing information with each other, (2) *Sharing*, where agents share information to model the openness together, and (3) *Informed*, where agents are given the degree of openness by the admin of the environment.

NoSharing. In our simulation model, let $\mathcal{L}_i$ denote the set of agent a_i's collaborators who have left the environment. Each agent a_i keeps track of their collaborators by storing its collaborators' information in a set $\mathcal{S}_i$, and checks the blackboard after every iteration for the information of the agents who have left the environment and updates $\mathcal{S}_i$. Agent a_i perceives the Agent Openness at time tick t using Eq. 5.1:

$$AO_i(t) = \frac{|\mathcal{L}_i|}{|\mathcal{S}_i|}\Big|_t \qquad (5.1)$$

Imagine a person who is working in a department of a big company. The company is so big that this person has no way of knowing the human resource changes of the entire company. The only piece of information this person can acquire is the changes of the employees in his/her own department, since these people are his/her coworkers (collaborators). This person can thus assess the company's human resource changes based on his/her own observation of his/her own department. If his/her coworkers get changed so often, then it makes sense to assume that the Agent Openness in this whole company is high and vice versa.

Sharing. In this case, agents will share the information of the agents with which they have worked to other agents in the environment and will share the "exited" agents that they have collaborated before in solving a task. Hence the Agent Openness can be perceived as in Eq. 5.2. Note that since now all agents share the same model, we do not denote AO with an underscript i, N_a is the total number of agents.

$$AO(t) = \frac{|\mathcal{L}_1 \cup \mathcal{L}_2 \cup \ldots \cup \mathcal{L}_{N_a}|}{|\mathcal{S}_1 \cup \mathcal{S}_2 \cup \ldots \cup \mathcal{S}_{N_a}|}\Bigg|_t \qquad (5.2)$$

Using the same example, if the person in the big company shares his/her colleagues' leaving information with people in other departments and get some information back from agents in other departments, then he/she would have a better idea of the change of personnel of the company as a whole.

Informed. The admin can also publish the AO on blackboard, so that every agent can know the exact AO and make the task selection decisions based on this true AO. This is represented in Eq. 5.3 as purely a constant value assignment.

$$AO(t) = AO \qquad (5.3)$$

Once again, using the same example, this is akin to the company announcing the number of employees leaving the company and the number of new employees joining the company.

5.3.4.2 **Perceiving Task Openness**

Task Openness (TO) is another important factor that affects an agent's judgment of expected utilities of solving a particular task—hence directly affect agents' decisions on selecting tasks. As mentioned Section 5.3.3.2, TO refers to the rate of new, previously

unseen tasks (task types) that are introduced into the environment, while previously seen tasks are retired. We provide three options for agents to perceive environmental Task Openness, similar to the options provided in perceiving Agent Openness:

NoSharing. An agent perceives TO base on the tasks it has seen by itself only, with no information sharing among agents. Let $\bar{\mathcal{T}}_i$ be the set of *task types* (Section 5.5) that one agent has seen, and let $\mathcal{N}_i$ be the set of tasks that one agent has encountered. In each iteration, when agent a_i checks the blackboard to see which tasks are available. Note that the tasks posted on blackboard consists of newly posted tasks and the reposted leftover tasks from last iteration's auction. The agent adds the task type $\mathcal{T}_{\text{type}}$ of each task T it sees to the set $\bar{\mathcal{T}}_i$. Note that it is a set, so if the reposted tasks have been previously added to this set, it will not be doubly counted. The cardinality of the set $\bar{\mathcal{T}}_i$, $|\bar{\mathcal{T}}_i|$, is the total number of distinct task types seen so far.

Meanwhile, the agent adds each task T to set $\mathcal{N}_i$. Note, again, the reposted tasks that have been seen by this agent before will not be counted again.

We use the ratio of cardinality of the set of distinct task types seen by one agent and total number of tasks seen by an agent to represent the perceived TO in Eq. 5.4.

$$TO_i(t) = \frac{|\bar{\mathcal{T}}_i|}{|\mathcal{N}_i|}\Big|_t \qquad (5.4)$$

For example, if an agent has seen 5 tasks in total, and 3 of them are distinct tasks (i.e., with different task types), then the TO should be $3/5$ =0.6. Thus, based on this perception, the agent can expect the next task it is about to see has 60% chance to have a new task type which is different from the ones it has seen before.

Sharing. Agents share information about tasks that they have seen and model TO together. In this case, agents will share the information of the task types as well as tasks they have seen so far to other agents in the environment. Hence the Task Openness can be perceived as in Eq. 5.5. Note that since now all agents share the same model, we do not denote TO with an underscript i.

$$TO(t) = \frac{\left|\bar{\mathcal{T}}_1 \cup \bar{\mathcal{T}}_2 \cup \ldots \cup \bar{\mathcal{T}}_{N_a}\right|}{\left|\mathcal{N}_1 \cup \mathcal{N}_2 \cup \ldots \cup \mathcal{N}_{N_a}\right|}\Bigg|_t \qquad (5.5)$$

Informed. In this case, the admin will publish the exact TO on blackboard, every agent has access to it, as shown in Eq. 5.6.

$$TO(t) = TO \qquad (5.6)$$

5.3.4.3 **Different Considerations for Perceiving AO and TO**

Note that there is a difference between the ways Agent Openness and Task Openness are defined: in AO, we do not consider agent types, but in TO, we do consider task types.

This is because we consider that every agent is unique in our model even when two agents have the same set of capabilities and each capability has the same quality. On the contrary, we consider that two tasks have the same task type if they consist the same set of subtasks and have the same task difficulty levels. (See Section 5.5.2)

The rationale behind this is that agents are capable of learning and their capabilities are changing dynamically as they live in the simulation environment. As a result, an agent's type changes over time. Hence, the makeup of agent types in the environment

changes accordingly as well. But as agents learn and evolve, these changes are to be expected and should not be considered as part of the agent openness.

Furthermore, agents also develop relationships with other agents they have worked with. When some agents leave the environment, the agents that are still in the environment will lose their relationships with those agents. If new agents with exactly same capabilities as those that left are introduced into the environment, they have no relationships with those agents already in the environment. Therefore, they are not considered as the same agents as those that left.

On the other hand, tasks will not change over time. The only thing that matters to the agents is task types. According to our definition of task types, a group of novice agents together could solve novice tasks. Similarly, a group of average agents are expected to solve moderate tasks, and a group of expert agents could solve hard tasks. Moreover, from the agent perspective, they simply treat the tasks that have the same task type as exactly the same task. For example, a task that has mopping the floor and cleaning the window as its two easy subtasks is no different that the other task that is comprised by same two easy subtasks with slightly different quality and/or number of agent requirements.

5.4 Agent Design (Agent Type)

In this section, we describe our agent design in detail. Section 5.4.1 lays out the design and lifecycle of the admin agent. Section 5.4.2 explains how we categorize agents into different types and the lifecycle of the individual agents in our model.

5.4.1 Admin Agent

The admin is the agent who is controlling the simulation environment. It peruses a configuration file for the simulation environment parameters. The admin serves three purposes.

First, it controls agents entering and leaving the environment based on the given agent openness (AO) parameter. Second, it discovers and decomposes task. Third, it holds auctions and allocates tasks. Note that, unlike agents (Section 5.4.2) that are able to solve tasks, the admin does not solve tasks.

There is no communication pathway between the admin and agents to share the admin's environment knowledge. Let A be the set of all agents in the environment, let $a_i \in A$ be an agent in A, to keep AO of the environment close to the given AO specification, the admin needs to periodically remove a set of agents A' from the environment and introduce the same number of new agents into the environment. The removed agents A' will be randomly selected from A. In the case that agent $a_i \in A'$ is busy doing a task T, the admin will remove it from the environment after a_i finishes the task immediately. The information of the removed agents is stored on the blackboard for existing agents to use to perceive agent openness in the environment.

The admin is also able to discover tasks from the environment through a domain-specific protocol. Let T denote a task that the admin has discovered. The admin can decompose the task T into a set of subtasks (recall that $T = \{\tau_1, \tau_2, \cdots, \tau_{|T|}\}$, and τ_k is a triple $\langle c_k, qt_k, n_k \rangle$), with the information of c_k as the skill or capability required, with its associated number of required agents n_k and the minimum threshold of quality qt_k

required of an agent in order to solve it. The decomposed tasks will be included in a message and be posted on Blackboard. After the admin posts tasks on the blackboard, it starts an auction session, as described in Section 5.6, Figure 5.2 below shows the lifecycle of the admin.

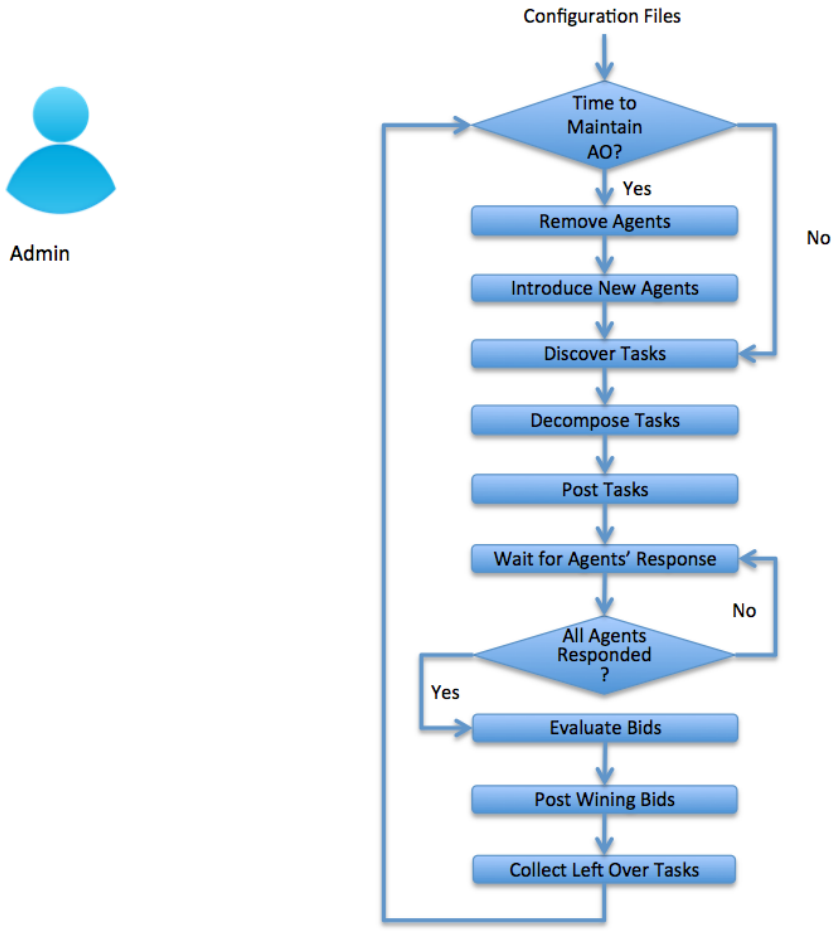

Figure 5.2 The lifecycle of the admin of the environment for our model. The arrows show the sequence of actions.

5.4.2 Individual Agents

Individual agents are the core task solving forces in our model. They have capabilities that correspond to the required skills to solve subtasks that are introduced into the environment. Moreover, each agent can improve their capabilities and update them dynamically.

We classify our agents into three types: (1) novice agent, (2) average agent, and (3) expert agent. We first define the capability type. Let $cap_{i,k} \in [0,1]$ be the quality of agent a_i's kth capability in $\boldsymbol{cap_i}$. We define $cap_{i,k}$ as novice capability if $cap_{i,k} \in [0.0, 0.3)$, $cap_{i,k}$ as average capability if $cap_{i,k} \in [0.3, 0.7]$, and $cap_{i,k}$ as expert capability if $cap_{i,k} \in [0.7, 1.0)$ (Table 5.1). Let $N = |c|$ (see Section 5.3.2) which is the cardinality of the set of all possible capabilities in the environment. Agents' types are classified based on Table 5.2. An agent is classified as Novice agent if $N/3$ or more of its capabilities are novice capabilities, less than $N/3$ of its capabilities are average capabilities, and less than $N/3$ of their capabilities are expert capabilities. In addition, an agent is called Average agents if less than $N/3$ of its capabilities are expert capabilities and $N/3$ or more of its capabilities are average capabilities. Moreover, we say an agent is an Expert agent if $N/3$ or more of its capabilities are expert capabilities.

Since our agents have learning ability, the capability types of their capabilities are changing dynamically during simulation. Hence one agent's type would change overtime and will be updated after its capabilities get changed. For example, a novice agent will be promoted to average agent once more than $N/3$ of its capabilities are

average capabilities after learning; a average agent will be promoted to expert agent once it has $N/3$ or more expert capabilities after learning.

Table 5.1 This table shows the classification criterion of agent's capability types.

Quality Range	$[0.0, 0.3)$	$[0.3, 0.7)$	$[0.7, 1.0]$
Capability Type	Novice	Average	Expert

Table 5.2 This table shows the classification criterion for the agent type based on the number of capabilities types of its capabilities.

	Number of easy capabilities	Number of average capabilities	Number of expert capabilities
Novice Agent	$\geq N/3$	$< N/3$	$< N/3$
Average Agent	*	$\geq N/3$	$< N/3$
Expert Agent	*	*	$\geq N/3$

An individual agent's lifecycle is shown in Figure 5.3. In each iteration, an agent starts with checking its status. If its current subtask on hand is not finished—i.e., it is busy, then it will keep executing the current subtask/subtasks. Otherwise, if it is not busy, not in middle of executing a subtask, then it checks the blackboard for new tasks as well as the published list of agents who have left the environment. The agent uses the information acquired to perceive the environmental Agent Openness (AO) and Task Openness (TO). Subsequently it uses this information to help analyze tasks based on task selecting strategies (Section 3.3.5). After that, each agent bids for one best task (if there is one, otherwise do not bid), submits the bid to blackboard auction and waits for the result. When the result is available, the agent checks the blackboard for task assignments. If it

wins the bid, it then starts executing its assigned subtasks. If it does not win the bid, it then goes to the next iteration.

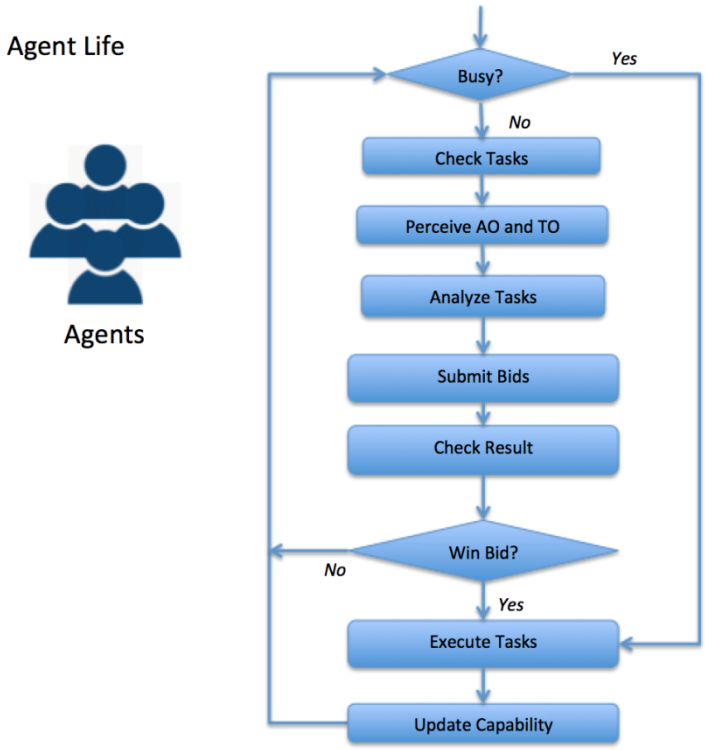

Figure 5.3 The lifecycle of individual agent of the environment for our model. The arrows show the sequence of actions.

5.5 Task Design

In parallel to the agent design, we also break tasks into three categories according to their difficulty levels. We have (1) easy tasks, (2) moderate tasks, and (3) hard tasks. Task difficulty levels are defined in Section 5.5.2.

When we consider whether two tasks, say tasks T_j and T_k, are of the same task type, they must satisfy two conditions. First, they must have the same set of subtasks. Second, T_j and T_k must have the same task difficulty level. The concept of task type is important, since it is tied to task openness and agents' perception of tasks openness.

5.5.1 Subtask Difficulty Level

In order to define task difficulty level, we first define subtask difficulty level. Subtask difficulty is defined in terms the quality threshold it requires as well as the number of agent it requires. Each subtask τ_k can be classified as easy subtask, moderate subtask, and hard task. We classify the difficulty level of subtask τ_k based on two parameters, one is the quality requirement qt_k , and the other is the number of required agents, n_k. Table 5.3 shows the classification criterion.

Table 5.3 This table shows the classification criterion for the difficulty level of subtasks. Note that $0<\beta<\alpha<1$; n_k denotes the number of required agents for solving a subtask; N_a is the total number of agents in the simulation environment. α, β, are parameters.

Subtask difficulty level	$1 \leq n_k < \beta N_a$	$\beta N_A \leq n_k < \alpha N_a$	$n \geq n_k N_a$
$0 \leq qt_k < 0.3$	Easy	Moderate	Hard
$0.3 \leq qt_k < 0.7$	Moderate	Moderate	Hard
$0.7 \leq qt_k < 1.0$	Hard	Hard	Hard

Notice that the classifier in terms of n_k is proportional to the total number of agents in the environment. We do it this way because the difficulty to find certain number of qualified agents to executing a subtask is proportional to the total number of agents, N_a, in the environment. For example, if we only have 50 agents in the environment, a subtask which requires 10 agents would certainly be a difficult one, but if we have 2000

agents in total, then it wouldn't be difficult any more. Instead, it would be a moderate, or even easy, subtask in this case.

As an example of this subtask classifier, in the case that we set the total number of agents $N_a = 200$, and set $\alpha = 0.015$, and $\beta = 0.01$, then we would have $\beta N_a = 0.01 * 200 = 2$, $\alpha N_a = 0.015 * 200 = 3$. Hence, a subtask which requires 3 or more agents is classified as hard subtask regardless of the quality requirements of the subtask, and a subtask requires 2 agents and has a quality requirement of 0.9 would also be a hard subtask. However, a subtask which requires 2 agents but the quality requirement is either 0.5 or 0.1, it is still be classified as a moderate subtask.

5.5.2 Task Difficulty level

With the definition of the difficulty of subtasks, now we define the difficulty of a task as follows.

Let the total number of subtasks in one task T to be N_T, let $\mathcal{E}_T$ denote the set of easy subtasks in T, $\mathcal{M}_T$ be the set of moderate subtasks in T and $\mathcal{H}_T$ stands for the set of hard subtasks in T respectively. For a task T, if the cardinality of $\mathcal{H}_T$ is greater than or equal to $\frac{N_T}{3}$, regardless of the value of $|\mathcal{M}_T|$ and $|\mathcal{H}_T|$, then we say this task T is dominated by hard subtasks, hence classify T to be a hard task. On the other hand, if a task T is not dominated by hard subtasks, i.e. $|\mathcal{M}_T| < \frac{N_T}{3}$, and the cardinality of $|\mathcal{M}_T|$ is greater or equal to $\frac{N_T}{3}$,then we say T is dominated by moderate subtasks regardless of the value of $|\mathcal{E}_T|$, and hence classify$T$ to be an moderate task. Finally, if a task T is

neither dominated by hard subtasks nor by moderate subtasks then we say T is an easy task (See Table 5.4).

For example, consider the tasks T_1, T_2, T_3, and T_4 in Table 5.5. We have $N_{T_1} = N_{T_2} = N_{T_3} = 5$, then $\frac{N_{T_1}}{3} = \frac{N_{T_2}}{3} = \frac{N_{T_3}}{3} = 2$. For T_1, we have $|\mathcal{H}_{T_1}| < 2$ and $|\mathcal{M}_{T_1}| \geq 2$, hence T_1 is a moderate task. For T_2 and T_3, we have $|\mathcal{H}_{T_2}| \geq \frac{N_{T_2}}{3} = 2$ and $|\mathcal{H}_{T_3}| \geq \frac{N_{T_3}}{3} = 2$, hence both T_2 and T_3 are classified to be hard tasks. Moreover, task T_4 has $N_{T_4} = 6$, $\frac{N_{T_4}}{3} = 2$, $|\mathcal{H}_{T_4}| < 2$, $|\mathcal{M}_{T_4}| < 2$, and $|\mathcal{E}_{T_4}| \geq 2$, therefore, T_4 is said to be an easy task.

Table 5.4 This table shows the classification criterion for the difficulty level of tasks. Here N is the total number of subtasks that comprise the task; $|\mathcal{E}_T|, |\mathcal{M}_T|$ and $|\mathcal{H}_T|$ are the number of easy subtasks, moderate subtasks, and hard subtasks respectively.

	$\|\mathcal{E}\|$	$\|\mathcal{M}\|$	$\|\mathcal{H}\|$
Hard Task	*	*	$\geq N/3$
Moderate Task	*	$\geq N/3$	$< N/3$
Easy Task	$\geq N/3$	$< N/3$	$< N/3$

Table 5.5 Examples of tasks that are classified as easy, moderate, and hard tasks. Using the task difficulty classifier, $\boldsymbol{T_1}$ is classified as moderate task, $\boldsymbol{T_2}$, $\boldsymbol{T_3}$ are hard tasks and $\boldsymbol{T_4}$ is an easy task

	T_1	T_2	T_3	T_4
$\|\mathcal{E}\|$	2	1	2	4
$\|\mathcal{M}\|$	2	2	1	1
$\|\mathcal{H}\|$	1	2	2	1
N	5	5	5	6
$N/3$	2	2	2	2
Difficulty Level	Moderate	Hard	Hard	Easy

We now give an example on determine whether the tasks are of the same type. Consider the tasks in Table 5.6 where tasksT_1, $T_2$and T_3 have the same set of subtasks, but each subtask has a different requirement for the minimum number of agents needed and a different quality threshold. Based on the task difficulty level classification criterion described in Table 5.4 and the subtask difficulty level classification criterion described in Table 5.3, given $N_a = 200$, $\alpha = 0.015$, and $\beta = 0.01$, we can classify T_1 as a moderate task, T_2 a hard task, and T_3 a moderate task. Hence T_1 and$T_3$are considered as having the same task type while the pairT_1 and T_2, and the pair T_2 and T_3 are considered to as having different task types.

Table 5.6 Example of tasks that can and cannot be considered to have the same task type T_1 is a moderate task, T_2 is a hard task, and T_3 is a moderate task. T_1 and T_3 are considered as having the same task type while T_1 and T_2, also T_2 and $T_3$are considered as having different task type. Note the subtask difficulty level is determined based on Table 5.3 with $N_a = 200$, and set $\alpha = 0.015$, and $\beta = 0.01$

T_1					
Subtask	τ_1	τ_5	τ_{13}	τ_4	τ_9
n	1	2	3	2	1
qt	0.2	0.4	0.3	0.5	0.5
Subtask Difficulty Level	Easy	Moderate	Hard	Moderate	Moderate
T_2					
Subtask	τ_1	τ_5	τ_{13}	τ_4	τ_9
n	2	3	1	1	1
qt	0.7	0.2	0.1	0.4	0.5
Subtask Difficulty Level	Hard	Hard	Easy	Moderate	Moderate
T_3					
Subtask	τ_1	τ_5	τ_{13}	τ_4	τ_9
n	2	2	1	1	2
qt	0.7	0.2	0.1	0.1	0.5
Subtask Difficulty Level	Hard	Moderate	Easy	Easy	Moderate

5.6 Blackboard and Auction Design

The center of our system is a blackboard-based publish-subscribe system (Wooldridge, 2009).This system provides a place for interacting and coordinating between agents in the environment, and provides information about current available tasks to agents. It has been proved that such design can eliminate the demands for explicit coordination and communication protocols between agents (Stone et al. 2010). In our design, tasks are allocated through an auction, which is held by admin through the blackboard. Figure 5.4 shows the timeline of admin and agents communicate and allocate tasks through blackboard. At the beginning of each iteration, the admin maintains Agent Openness (AO) by removing and introducing agents. The information of removed agents is stored on the blackboard by the admin. Then the admin posts a list of messages on the blackboard, which contains one new task that the admin chose from the task pool as well as the tasks that have not been auctioned off. Agents in the environment can check these messages to see current available tasks. Then the admin starts an auction session for all tasks on the blackboard. If an agent is not busy (idle), then, after perceiving and updating AO and TO of the environment by accessing the stored list of removed agents as well as the available tasks on blackboard, each agent analyzes current available tasks on the blackboard and bids for the one that returns the highest potential utility. To analyze current available tasks on the blackboard, an agent adheres to a certain task selection strategy (we have designed several strategies in Section 3.3.5) and bids for at most one task in one iteration by submitting the bid to the blackboard. After that, the admin gathers all the bids, allocates the task using a task allocation algorithm (described in the next paragraph) to assign each task to the best capable agents who bid for the task, and posts

the auction results on the blackboard. Upon the disclosure of the auction results, agents will be notified and checking back on the blackboard. Winning agents will then start to execute the subtasks to which they are assigned and other agents will wait for the next iteration.

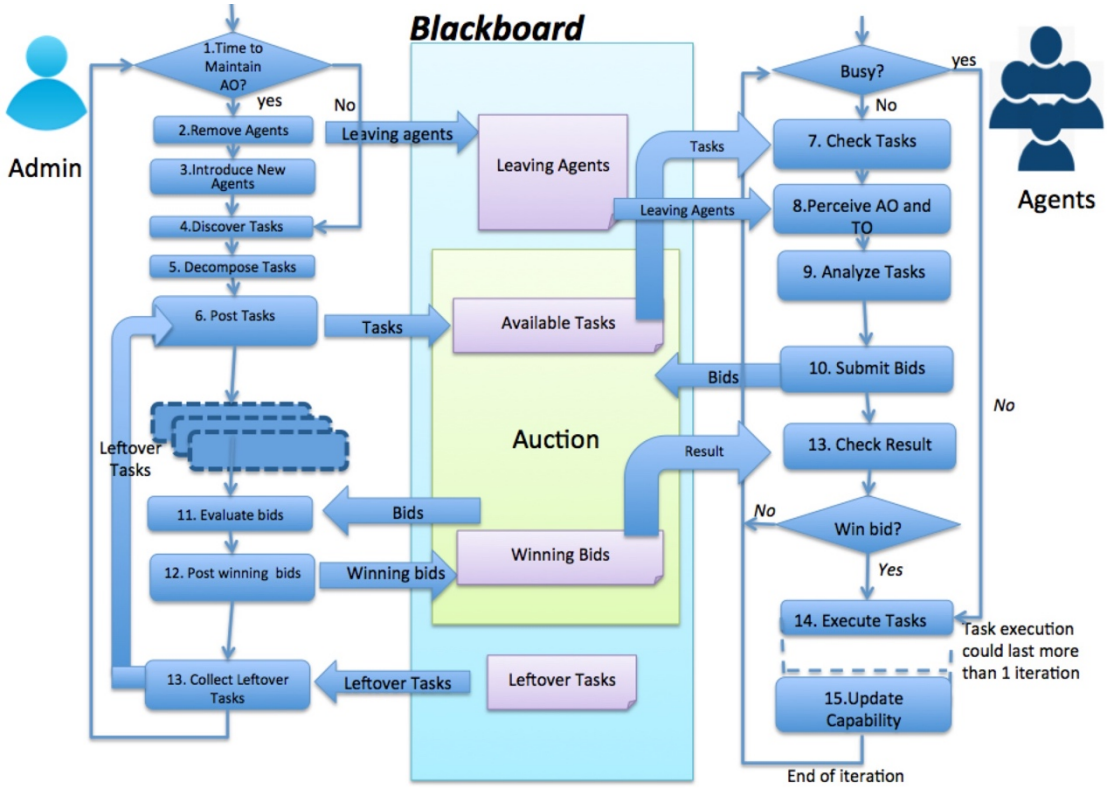

Figure 5.4 Admin and agents communicating and allocating tasks through blackboard. The arrows show information flows between the admin and blackboard as well as those between agents and blackboard. The sequence of actions is designated as well.

During the auction, let A_T denote the set of agents that bid for task T. For each subtask$\tau_k \in T$, the admin selects the top n_k agents $a_i \in A_T$ that have the highest capability $cap_{i,k}$ such that $cap_{i,k}$ is larger than the quality threshold qt_k. If at least one subtask fails to have enough qualified agents, then the whole task will fail to be auctioned

off. Algorithm 5.1 below shows the details of the auction algorithm the admin agent uses to allocate tasks.

Start Algorithm *Auction* (Blackboard *b*)

1. Set $\mathcal{M} \leftarrow allMessagesOnBlackboard$
2. **Foreach** message m_i **In** $\mathcal{M}$ **Do**
4. $S \leftarrow \{\emptyset\}$ // set with assignment pair (a_i, τ_k)
5. $T \leftarrow The\ task\ contained\ in\ m_i$
6. $A_T \leftarrow \{a_i |\ a_i \in all\ agents\ bidding\ for\ T\}$
7. InnerLoop:
8. **Foreach** subtask τ_k **In** T **Do**
9. Sort $A_T$base on agents' quality of τ_k from high to low
10. **If** $|A_T| >= n_k$ **Then**
11. Let $a_j \leftarrow$ the n_k th agent in A_T
12. **If** $cap_{j,k} > qt_k$ **Then**
13. **For** *i* from 1 to n_k **Do**
14. Add the pair (a_i, τ_k) to the assignment S
15. **End**
17. **Else**
18. Post m_i to *b.returnedMessage* //Main agents may introduce this task again in the future ticks.
19 Break InnerLoop
20. **End**
21. **Else**
22. Post m_i to *b.returnedMessage*
23 Break InnerLoop
24. **End**
25. **End**
26. Post assignment S to *b*
27. Remove m_i from Blackboard
28. **End**

End Algorithm

Algorithm 5.1 Auction algorithm used by admin to allocate tasks

5.7 Probabilistic Model

In this section, we talk about the probabilistic model we used in two of our included task selection strategies in detail. These two task selection strategies were used in Chen et al. (2016).

As mentioned in Section 3.3.2, due to the openness, there are uncertainties in the task assignment. An agent who bids for a task may or may not get the task assignment due to two reasons: (1) it does not win the bid since the admin agent only chooses top n_k bidders for each subtask τ_k) and (2) there are not enough agents with qualified capabilities bidding on the task to form a collaborative team. Hence, we use two probabilities to estimate the uncertain task assignment. Recall that $P_{wb}(\mathrm{T})$ is the probability that the agent can win a submitted bid (i.e., the agent is one of the top n_k bidders for some subtask τ_k). $P_{off}(T|wb)$ is the probability that the task will be auctioned off (i.e., enough agents with qualified users bid on the task to form a collaborative team), conditioned on the event that the agent wins the bid.

For each task, the admin agent will disclose the auction result immediately after the auction. The result is contained in two hash maps: (1) *subtaskWinningAgentMap* and (2) *subtaskAssignmentMap*. Both hash maps map the subtask id to an arraylist of agent ids. When examining the *subtaskWinningAgentMap*, an agent finds the arraylist using the subtask id of the subtask it bid on, and then tries to find its id in this arraylist. If its id is found in the list, then the agent won the bid (i.e., it was ranked the as the top n_k bidders, and was selected for performing the subtask). Otherwise, it lost the bid. Notice that winning the bid does not guarantee the agent can get the subtask. Whether the task can be auctioned off or not depends on if enough qualified agents for each subtask can be found. If one subtask fails to have enough number of qualified agents, then the whole task fails to be auctioned off. In the case that the task fails to be auctioned off, arraylists in the *subtaskAssignmentMap* will be empty. Agents examine the *subtaskAssignmentMap* in the same way to find out if they get the subtask assignment or not.

In order for an agent to learn $P_{wb}(T)$, it stores the tasks it has ever attempted (the bids) with the bidding results (i.e., (1) whether it won the bid and (2) whether it was assigned the subtask) found in *subtaskWinningAgentMap* and *subtaskAssignmentMap* in *agentbidingList*. Then the agent computes the Euclidean distance between the task T and all the tasks in the *agentbidingList* using the qt_k(the quality threshold this subtask τ_k requires and n_k (the minimum number of qualified agents this subtask τ_k requires) values in subtasks $\tau_k \in T$ to find the most similar *s* tasks. In our simulations, we set $s = 5$ and simulation users can change this value accordingly to fit their research needs. We can now apply Eq. 4.18 in Section 4.4.3 to find $P_{wb}(T)$

$$P_{wb}(T) = \frac{1}{|S(T)| + \epsilon'_{wb}} \sum_{T' \in S(T)} won(T') + \epsilon_{wb}$$

here $S(T)$ is the s-most similar tasks that the agent previously bid on, $\sum_{T' \in S(T)} won(T')$ gives the count of tasks she won among $S(T)$, $\epsilon_{wb} = 1/|S(T) + 1|$ and $\epsilon'_{wb} = 4/|S(T) + 1|$.

Similarly, we can calculate $P_{off}(T|wb)$ using the Eq. 4.19 in Section 4.4.3

$$P_{off}(T|wb) =$$

$$\frac{1}{\sum_{T' \in S(T)} won(T') + \epsilon'_{off}} \sum_{T' \in S(T)} auctionedOff(T') + \epsilon_{off}$$

where $\sum_{T' \in S(T)} auctionedOff(T')$ gives the count of tasks that she won and also auctioned off, $\epsilon_{off} = 1/|S(T) + 1|$ and $\epsilon'_{off} = 4/|S(T) + 1|$.

Table 5.7 shows how we compute each component of the above two equations:

Table 5.7 Methods of calculating components in Eq. 4.18 and Eq. 4.19

Component in equation	Methods of computing
$\lvert S(T)\rvert$	Compute the Euclidean distance between the task T and all the tasks in the *agentbidingList* using the qt_k and n_k values in subtasks $\tau_k \in T$ to find the most similar s tasks. We set $s = 5$ in our simulations, hence $\lvert S(T)\rvert = 5$.
$\sum_{T' \in S(T)} won(T')$	Go over the *agentbidingList* and sum over the bidding results for the tasks $T' \in S(T)$. Recall that *agentbidingList* stores all the bidding history, including the bidding results and task information.
$\sum_{T' \in S(T)} auctionedOff(T')$	Similar to the method for finding $\sum_{T' \in S(T)} won(T')$. Go over the *agentbidingList* and sum over the bidding results for the tasks $T' \in S(T)$. If the bidding result for the task is marked as auctioned off, then add 1 to the *Sum*.

5.8 Learning

We focus on two types of learning in our simulation, learn by doing and learn by observation. Our simulator has two implementations of both learning type, and they are similar in nature but are based on different research papers.

The first implementation was used in Chen et al. (2015) and is discussed in detail in Section 3.3.4. The *learning by doing* algorithm is based on Jumadinova et al., (2014). The *learning by observation* algorithm was given by us and is based on Vygotsky's zone of proximal development (ZPD) theory (Vygotsky, 1978). We include the equations and code snippet for this implementation below.

The following equations are from Section 3.3.4, Eq. 3.1 and Eq. 3.2 were used for calculating agent $a_i's$ capability gain on capability k and for calculating the capability gain for agent a_l observing agent a_t successfully completing a subtask k respectively:

$$Gain_{self}(a_i, k) = \frac{\eta}{cap_{i,k} + \varepsilon}$$

where η is a constant denoting the increment in knowledge from self-learning and ε is a small number in case $cap_{i,k}$= 0.

$$Gain_{Observe}(a_l, a_t, k) = \begin{cases} 0 & if\ x < 0 \\ -\frac{\beta}{\alpha^2}x^2 + 2\frac{\beta}{\alpha}x & if\ 0 \le x < \alpha \\ -\frac{\beta}{(\alpha-1)^2}x^2 + \frac{2\alpha\beta}{(\alpha-1)^2}x + \frac{\beta(1-2\alpha)}{(\alpha-1)^2} & if\ \alpha \le x < 1 \end{cases}$$

where x is the capability difference between agent a_t and agent a_l, $x = cap_{t,k} - cap_{l,k}$ and β is the maximum learning gain that a_l can acquire from observing agent a_t, and α is the capability difference that gives the maximum learning gain. The following algorithm (Algorithm 5.2) was used for agents to calculate the learning gains and update its capabilities after the completion of the task in Chen et al. (2015).

Start Algorithm *UpdateCapabilities* ()

1. $m \leftarrow MessageTheAgentBidfor$
2. $T \leftarrow The\ task\ contained\ in\ m$ (note that $\boxplus \{\tau_1, \tau_2, \cdots, \tau_{|T|}\}$)
3. $T' \leftarrow \{\tau_k | \tau_k \in T\ and\ agent\ participated\ in\}$
4. $T'' \leftarrow T \setminus T'$ //Set of subtasks the agent observed
5. **Foreach** subtask $\tau_k \in T'$ **Do** //calculate learning by doing gain
6. $a_l \leftarrow agent\ itself$
7. **If** $cap_{l,k} + Gain_{self}(a_l, k) > 1$
8. $cap_{l,k} \leftarrow 1$
9. **Else**
10. $cap_{l,k} \leftarrow cap_{l,k} + Gain_{self}(a_l, k)$
11. **End**
12. **End**
13. **Foreach** subtask $\tau_k \in T''$ **Do** //calculate learning by observation gain, only learn from the one who gives the max observing gain
14. Set $A \leftarrow$ agents assigned for τ_k
15. $MaxGain \leftarrow 0$
16. **Foreach** $a_t \in A$
17. **If** $Gain_{Observe}(a_l, a_t, k) > MaxGain$
18. $MaxGain \leftarrow Gain_{Observe}(a_l, a_t, k)$
19. **End**
20. **End**
21. **If** $cap_{l,k} + MaxGain > 1$
22. $cap_{l,k} \leftarrow 1$
23. **Else**
24. $cap_{l,k} \leftarrow cap_{l,k} + MaxGain$
25. **End**
26. **End**

End Algorithm

Algorithm 5.2 Algorithm for calculating the learning gains and updating capabilities

The second implementation was used in Chen et al. (2016) and is discussed in Section 4.3. In this implementation we use the exponential learning equation for success-based learning (Leibowitz et al., 2010) for learning by doing:

$$\Delta_{do} cap_{h,k} = \dot{cap}_{h,k} = \alpha_{do} \cdot cap_{h,k} \cdot \left(1 - cap_{h,k}\right)$$

The learning by observation algorithm was based on Bandura's social cognitive learning theory (1986, 2004):

$$\Delta_{obs} cap_{h,l} = \begin{cases} \dot{p} & 0 \leq qt_l - cap_{i,l} < \beta \\ 0 & otherwise \end{cases}$$

where β is the threshold under which $qt_l - cap_{h,l}$ is small enough for learning by observation to take place and

$$\dot{p} = \alpha_{obs} \cdot \left(qt_l - cap_{h,l}\right) \cdot \left(\beta - \left(qt_l - cap_{h,l}\right)\right)$$

Agents uses an algorithm very similar to Algorithm 5.2 to update their learning gains, except (1) now they use the above equation variants to calculate gains from learning by doing and learning by observation, and (2) they *only* observe the subtask—instead of all subtasks performed by teammates—that gives the maximum learning by observation gain.

5.9 Configurable Parameters

There are different configuration parameters that can be changed, which enables different type of experimentation and tests within the multiagent ad hoc team formation problems. All the configuration parameters are discussed in remaining portion of this section. Table 5.8 shows a summary of the configurable parameters as well as their value ranges.

Table 5.8 Configurable parameters and their value ranges

Parameters Category	Parameter Specifications
Subtasks Configuration for individual tasks.	Number of Subtask >=1subtask, Minimum Quality Threshold per Subtask: [0.0 - 1.0] Minimum Number of Agent Required per Subtask >=1agent
Agent Makeup Configuration	Percentage of Agent Type : [Any combination adding up to 100%]
Environmental Openness	Agent Openness: [0.0 – 1.0] Task Openness: [0.0 – 1.0]
Task Selection Strategies	Built-in Strategies : [1-10] Custom Strategies : As many as required
Simulation Length	>= 1 tick
AO/TO Perception	Sharing, No Sharing, and Informed
No. of Initial Non-Zero Capabilities	>= 1capbability
Tick to Finish	>= 1tick
Number of Agents	>= 1agent
AO implementation	1 or 2
TO implementation	1, 2 or 3

5.9.1 Subtasks Configuration for Individual Tasks

The numbers and variety of tasks within the simulation environment are configurable. Tasks configuration includes all the properties of tasks like total number of tasks available, and distribution of tasks difficulty. There is also a more granular control, where the properties of each task within a task pool can also be modified. For example, each task can be modified in terms of (1) the number of subtasks in a task, (2) the minimum quality threshold that agents are required to solve a subtask, and (3) the minimum number of agents required to solve each subtask. This configuration allows

researchers to modify tasks based on the environment they are modeling which can contain tasks which are almost identical to very different, or tasks which are very easy to solve to very difficult to solve.

5.9.2 Agent Makeup Configuration

The number and types of agents can also be configured in MAAHTFormS. This configuration allows simulation to contain different mixtures of expert, average, and novice agents. Moreover, agents can be generated based on certain mathematical/statistical distributions, allowing researchers to create simulation with truly varied agent makeup. In order to track agents' activities throughout the simulation, some agents can be configured to not be removed from the simulation at all. This configuration parameter can thus control how many skills agent can have initially, what sort of agent mixtures the environment can have, etc.

5.9.3 Environmental Openness Configuration

MAAHTFormS also allows the control of both agent and task openness. Thus it is possible to set agent and task openness to any value between 0.0 to 1.0 and experiment with different sets of openness values.

5.9.4 Task Selection Strategies

Based on the weights given to learning and solving tasks, MAAHTFormS technically allows an infinite number of task selection strategies, with strategies using AO only, TO only and both AO and TO, or neither. We provide some examples of task selection strategies in Section 3.3.5, which used different weight combinations for

learning and solving tasks, such as $w_L = 0.25$ and $w_S = 0.75$, $w_L = 0.5$ and $w_S = 0.5$ etc. In all cases, the weights of learning and solving tasks sum to exactly 1. This configuration parameter allows abstracting different task selection strategies based on how agents will pursue immediate vs. future task rewards. We also include some task selection strategies that is used by Chen et al. (2016) from Section 4.4. By mapping those choices into the weight parameters for solving and learning task, creating new task selection strategies becomes an easy task.

5.9.5 Simulation Length

This parameter controls the length of the simulation, which makes it possible to perform simulation for different length of time. The unit for the simulation is "tick", which is one cycle of operations. For example, in our simulation, when all individual agents finish their cycle once (Section 5.4.2) and the admin agent finishes its cycle once (Section 5.4.1), that is the end of 1 tick. In ad hoc teams, a short simulation length might prevent agents from effectively utilizing the capability they learn whereas a longer simulation might allow them to actually use their gained capabilities. Some emergent behaviors might need longer simulation lengths, e.g. 1000 ticks, to be observed. So, this parameter can be modified and changed, to study agent behavior, overall system behavior etc. for different period of time.

5.9.6 AO/TO Perception Configuration

Since we have implemented different options to perceive openness as described in Section 5.3.4, this configuration enables selecting different mechanisms to perceive openness by the agents. Here we briefly summarize these perception options.

"Sharing" allows agents to share their observations on the agents they worked with (for agent openness) or on the tasks they encountered, e.g. the tasks they saw on blackboard as well as the tasks they solved (for task openness). This option makes the modeling of openness a team effort and every agent has the same openness perception, since they all have the same information.

In "No Sharing", agents keep information to themselves. There is no communication between agents about their observations, agents model openness based on their entirely own observations. This option usually results in agents having inaccurate openness perceptions, because each agent has limited observations of the entire environment.

The "Informed" option allows agents be given the actual openness instead of agents modeling openness themselves.

Depending upon what kind of agents and environment is being modeled, it makes sense to have different perception models for openness, as some times agents might implicitly share some of the openness information to the other agents in the system whereas other times there might be no communication, implicit or explicit.

5.9.7 Number of Initial Non-Zero Capabilities

Since the number of initial non-zero capabilities represents how many different types of capabilities an agent possesses before the start of the simulation, this parameter models the overall initial capability of the entire agent population. This parameter can be configured to represent a whole spectrum of agent capability makeup from agents which are very knowledgeable at the beginning of the experiments to agents which know next to

nothing. If the number of initial non-zero capabilities is too small, e.g., 1, a lot of tasks might not be able to be solved due to lack of expertise in the environment. However, too much expertise in the environment (the number of initial non-zero capabilities is too large, e.g., 20) will increase the competition among agents.

5.9.8 Tick to Finish

In our current design, all the tasks in the environment use the same number of ticks to finish. Tick to Finish refers to the number of ticks it requires to finish each task. The significance of this parameter is that if an agent is involved in completing a task, then it is not allowed to bid for another task or subtask. So, the longer the value is, the fewer idle agents are available in the environment at each tick, and vice versa.

5.9.9 Total Number of Agents (N_a)

This parameter specifies the total number of agents in a simulation at each tick. Based on the AO parameter, old agents are removed and new agents are added by the auctioneer (admin) during the simulation; but the total number of agents, at each tick, is always N_a.

5.9.10 AO/TO implementation

We include two AO/TO implementations for users to choose from. The first AO/TO implementation is used in Chen et al. (2015) and the second AO/TO implementation is adapted by Chen et al. (2016). Additionally, a third TO implementation is also available for users to choose from. The details of all AO/TO implementations can be found in Section 5.3.3 of this chapter.

5.10 Data Generated from the Simulator

We log our simulation outputs and then run a program to do the analysis and produce a report. We mainly focus on logging individual agent's behavior such as what task it bid on, whether it won the bid or not, and whether it get assigned any subtask etc. Table 5.9 shows the variables we log.

Table 5.9 The variable values logged and its description for the simulation

Variable Name	**Description**
tick	The current tick of the simulation
id	The id of the agent
numTaskInvolved	The total number of tasks that this agent get assigned during the simulation
taskAssignmentAtCurrentTick	The task id of the task that this agent was assigned to at current tick. If no assignment, this number will be 0
reward	The task reward this agent received for help finishing the current task
taskbidAtCurrentAtCurrentTick	The task id of the task that this agent bid on at current tick. If agent does not bid, this number will be 0
numBidsSumitted	The total number of bids this agent submitted during the simulation
numBidsSumbittedAndWon	The total number of winning bids during the simulation
selectedForCurrentBid	A Boolean value indicating whether this agent won the current bid or not
taskTypeBidOn	The task type id this agent bid on at current tick
randomSeed	The random seed for current simulation
agentOpenness	The agent openness for current simulation
taskOpenness	The task openness for current simulation
option	The task selection strategies this agent is using in current simulation
numAgentsAssigned	The total number of agents the current task (the task this agent bid on) uses, if this task did not get

	auctioned off, the number will be 0. Notice: one agent could get multiple subtasks. Hence this number does not always equal the total number of agents the current task needed
numAgentsRequired	The total number of agents the current task (this agent bid on) needed
taskReward	The task reward of the current task (this agent bid on)
selfGain	The capability gained through doing the subtask
observationGain	The capability gained through observing its teammates

The log files we get from simulation are very large due to the number of variables we log. For one configuration of a simulation, the log file is about 1 MB if we choose the number of ticks to be 100. If we choose the number of ticks to be 1000, then the size of the log file is about 10MB. Sometimes we do need to set the number of ticks to 1000 or even bigger number to see the emergent behavior. For a full simulation, which means includes every agent openness, task openness, and option combination, we will get as many as 5880 log files. (we set *AO/TO*= [0,0.01,0.02,0.05,0.1,0.2,0.5], *Option*= [1,2,3,4] and use 30 random seeds, hence $7\times7\times4\times30 = 5880$). Therefore, for a 100-tick full simulation, our log files size could be almost 6 GB, and near 60 GB for a 1000-tick full simulation. The size of large log file is a problem and created a challenge for us, since we do thousands of such simulations, and will run into storage problems very quickly. We noticed that some of the variables has the same value throughout the simulation hence it makes sense to exclude them from the log file and put them into the file name to reduce log file size. Such variables include *RandomSeed, AgentOpenness, TaskOpenness*, and *Option*. After such change, our log file name looks like "AgentOutput_AO[x]TO[y]Op[z]_[timestamp]", where "x" is the agent openness value,

"y" is the task openness value, "z" is the task selection strategy number that all the agents use during the simulation, and the "timestamp" is the date and time the simulation started. In addition to that, we also included a boolean variable called "agentOutputShort" in OutputClass.java. If it's value is set to be true, then we only log the variable values in Table 5.10.

Table 5.10 Variable values logged and its description for the simulation when "agentOutputShort" is set to be "ture"

Variable Name	**Description**
tick	The current tick of the simulation
id	The id of the agent
taskAssigned	Whether this agent get assigned to a task or not. 1 means has an assignment, 0 means not.
bidsWon	Whether this agent's bid won or not. 1 for yes, 0 for no.
taskReward	The task reward for the task this agent bid on
rewardGot	The task reward this agent got after finishing this task it bid on
selfGain	The capability gained through doing the subtask
obsGain	The capability gained through observing its teammates

After taking the above actions for reducing the log file size, we successfully reduces the log file size down to 400 KB for a 100-tick simulation, and about 4 MB for a 1000-tick simulation. Hence for a 100-tick full simulation, our log files size would be 6 GB, and for a 100-tick full simulation, the files size is about 2.4 GB, and 24GB for a 1000-tick simulation. After running the analyzing program, we zip the log files for archive purposes. The zip process reduces the file size more than 90%, and saves us a lot of resources.

5.11 Scripts for Running on Super Computer

We utilize the Holland Commuting Center (HCC)'s super computer to run our simulations. The documentations on how to use HCC super computers can be found on https://hcc-docs.unl.edu/display/HCCDOC/HCC+Documentation.

Our simulation program is written in Java, hence we make a jar file and put it on the super computer. We can run our program by executing the following line in terminal window:

```
"java -jar [jar file name] [AO] [TO] [your properties file]
[output directory]"
```

where "jar file name" is our simulation jar file's name, "AO" is the agent openness value, "TO" is the task openness value, "your properties file" is the properties file that contains all the task information (our task pool), and "output directory" is the directory where user wants to store the output log files. The other parameters are all set in Parameters.java file. We set the program to take the arguments this way in order to take advantage of HCC's super computer.

In order to use the HCC's super computer, we have to make a SLURM file and submit the SLURM job. The following code snippet in Figure 5.5 shows an example SLURM file we used for one part of our simulation. Notice that we can submit as many SLURM files as we need, hence we can split our simulations into many SLURM jobs to let them run simultaneously. To submit the SLURM file, we just type "$sbatch [filename].slurm", where "filename" is your SLURM file name.

```
#!/bin/sh

#SBATCH --time=3:00:00          # Run time in hh:mm:ss

#SBATCH --mem-per-cpu=8G          # Maximum memory required per CPU (in megabytes)

#SBATCH --job-name= “20cap_50_50”

#SBATCH --error=/work/soh/bchen/ad-hoc/20capAO0.5TO0.5.err

#SBATCH --output=/work/soh/bchen/ad-hoc/20capAO0.5TO0.5.err

module load java

java -jar ad-hocOp1.jar 0.5 0.5 20choose5.properties /work/soh/bchen/ad-

hoc/20cap/20cap1000tick
```

Figure 5.5 Sample SLURM file we used in part of our simulation

Chapter 6: Conclusions and Future Work

In this thesis, first, we have developed an auction-based multiagent simulation framework to study/investigate the impact of Agent Openness (AO) and Task Openness (TO) in an multiagent task execution scenario. The Java-based simulator that we have developed is called Multi Agent Ad-Hoc Team Formation Simulator (MAAHTFormS). We conducted comprehensive experiments, established the importance and necessity of considering AO and TO in ad hoc team formation problem, and also discovered the impact of AO and TO on agent learning and task completion under varying degrees of environmental openness in our task execution scenario. We considered the aspect of agents learning and evolving, and proposed several agent task selection strategies to leverage the environmental openness. Our study has gained insights into the relationships between AO and TO. We have seen that AO and TO change the way teams are formed in ad hoc setting. When making decisions on which teams to join, agents should consider the possibility of new agents and tasks entering the environment. Furthermore, we have seen that AO impacts learning. AO is helpful to boost the learning when new tasks appears in the environment as new tasks requires new capabilities to solve. We have also seen that TO makes tasks more challenging for agents to solve.

Second, we have studied an agent-based collaborative human task assignment problem, which is a direct application of ad hoc team formation problem in open systems. We have developed solutions for agents to maximize their users' rewards and learning gains over sequence of tasks under the environmental openness (AO and TO). More specifically, we developed a probabilistic model, which agents learns about to guide its

decision making in maximizing human user reward and learning gains and we modeled human learning and incorporated it into the agent's reasoning on how to acquire tasks for its user. we have shown through empirical experiment that our Uncertainty and Learning-Aware (ULA) agents are capable of choosing tasks maximizing expected utilities taking into account the uncertainties and learning.

Third, we have developed the aforementioned simulator called Multi Agent Ad-Hoc Team Formation Simulator (MAAHTFormS), which can be used for very comprehensive multiagent ad hoc team formation simulation. It simulates the task openness and agent openness which can be used to analyze and understand the interrelationships between several important factors in the realm of this problem. More specifically, MAAHTFormS allows researchers to study team formation in an open environment, to study develop and test task selection strategies while considering openness, and to study the impact of diversity, among many other things.

6.1 Future Work

6.1.1 Immediate next Steps

a. Find better ways to model AO

In our current framework, agents model both agent openness and task openness. Considering both types of openness, the idea was for an agent to develop more effective task selection strategies to better leverage them. However, we were only able to get good modeling for task openness. The model for agent openness is not quite accurate so far. This is because agents can observe the blackboard for task information so they can have

very good ideas about the newly listed tasks as well as the tasks that disappeared. The only time that an agent will miss the task information is when it was executing tasks. Hence the model for task openness is very close to the actual value. However, agents only know other agents through their collaborations in task solving. Due to lack of pre-coordination and the limited agent information, the perceived agent openness was far off from the actual value. For the proposed task selecting strategies, agents are given the AO and TO. We will explore more realistic ways to perceive openness, such as (1) NoSharing, where agents model on their own without sharing information with each other, (2) Sharing, where agents share information to model the openness together. This will be a key next step to take in the future, since sensing the environment and making autonomous decisions are the fundamental functions of agents. We give agents the AO and TO information in our current research to simplify the complicity of this problem as our first step to investigate the impacts of the AO and TO in ad hoc team formation.

b. Different task assignment policies

Our current simulation framework is auction-based. The auctioneer collects all the bids and assigns the best agents for the tasks. If an agent does not get the task it bids for, then the agent does not perform any work and will have to wait until the next auction round. In a more realistic scenario, this agent may still have other skills which can be used to team up with other agents who also do not get assigned tasks, and together could accomplish some other tasks. We can explore new task assignment policies to utilize this under-utilized workforce at each time tick. We believe with the new task assignment policies, there will be more tasks solved in unit time (per tick per agent) and more

learning will occur. This might counter some negative impact of TO, since there will be more expertise can be utilized in the system.

c. Different bidding protocols

In our current design, an agent only bids for one best task in the auction according to the algorithm with which it is deployed. We can explore other bidding protocols such as allowing agents to submit multiple bids for a single auction. For example, the algorithm can give the ranks of the preferred tasks, and the system will then allow agents to have first preferred bid, second preferred bid, etc. We have already seen that there are lots of tasks did not get auctioned off due to the competition. Many agents try to bid on the same task but the auctioneer only chooses the best required number of agents for the task. In this case, lots of agents who lost the bid got no task to do for that round of auction. This results in good expertise get wasted. Since there were many tasks that the agents who lost bids are more than capable of, if they have bidden on these tasks, they could have gotten the task. We believe a well-designed new bidding protocol could help ease this phenomenal of expertise waste due to the completion, and hence boost the system performance in general.

d. Consumption of time on tasks

As mentioned in Chapter 5, our current implementation assumes every task takes the same time, which is 1 tick, to be completed. This is not quite realistic as in the real world different tasks have different levels of complexity, and they consume different resources including time. It is obvious that simpler tasks can be done faster and harder tasks need more time to be completed. This should be considered in the simulation to

make it more realistic. However, this will add a lot of complexity as well into the problem of estimating the long term rewords, since sometimes completing several easy tasks can gain more immediate rewards than spending more time on complex tasks, but the complex tasks may gain the agent more potential rewards if the task have more learning opportunities.

6.1.2 More "further" next Steps

In terms of further next steps, we have several considerations.

a. Consider teaching

We will consider the impact of *both* teaching and learning while modeling agent's behavior, particularly incorporating the fundamental game-theoretic work from Stone, Gan, & Kraus (2010). This will require agents to consider the potential gain from teaching another agent, as opposed to only considering potential gain from learning from other. Indeed, by teaching other members in the team can gain the teacher agent long term reward when the learner agent can stay long enough to implement and improve the team's utility with what it has learned. Hence, agent openness is a key factor for the teacher agent to make decisions on whether to teach or not teach. If the learner agent will leave shortly, teaching would not be beneficial. Instead, the teaching agent will be better off by improving its own expertise or to complete more tasks to gain immediate rewards.

b. Consider agent reliability

We will *consider agent reliability* in terms of agent possibly failing to complete tasks. This can be built into agent reasoning when making the bids for tasks as part of

solution robustness consideration. Agents have little or no knowledge of the capabilities of other agents in the ad hoc team formation environment, agents can build trusts among themselves. In addition, agents can build its reputations of being reliable, or not accountable. When we allow agents to submit multiple bids, if a potential teammate is not accountable, then instead of risking the task, a better choice will be to decline the bid and wait for the results of other bids. In agent reasoning, when an agent try to select a task to bid for, the best potential utility is based on also a probability of successfully executing the task. We can include a modeling of agent reliability in terms of (1) weather the potential teammate will be able to carry out the subtask assigned to them successfully, (2) whether the potential teammates will accept the subtasks. This will make agents smarter in terms of agent reasoning, and make our system more robust.

c. Investigate diversity

We plan to study the impacts of the amount of diversity in the task types and in the agents' capabilities. For example, if the environment only has a small set of highly capable agents/human to begin with, will the learning be able to counter the impacts of openness? How many diverse expertise will be good enough for the system in the current open environment to successfully deal with the openness? Diversity in agent/human expertise can affect how the system adapts. Likewise, diversity of task types can affect how agents/human learn and their ability to complete tasks. Too much diverse expertise in agent population can cause agents to spread their bids to much so that only a few tasks can be auctioned off or completed as they try to maximize their long-term utility to become more qualified in more different tasks. Maybe agents should consider when to learn and when not to learn based on how diverse the tasks are as well as how diverse the

agents are. In addition, agent may need to decide on whether to be an expert in one area, or to learn more skills to be a generalist.

References

Agmon, N., Barrett, S., & Stone, P. (2014). Modeling uncertainty in leading ad hoc teams. *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems.

Agmon, N., & Stone, P. (2011). Leading Multiple Ad Hoc Teammates in Joint Action Settings. *Interactive Decision Theory and Game Theory*, 2–8. Retrieved from http://www.aaai.org/ocs/index.php/WS/AAAIW11/paper/viewPDFInterstitial/3805/4277

Ahn, L. Von, Maurer, B., Mcmillen, C., Abraham, D., & Blum, M. (2008). reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, *321*(12 September 2008), 1465–1468. https://doi.org/10.1126/science.1160379

Bandura, A. (1986). *Social foundations of thought and action: A social cognitive theory*. Prentice-Hall, Inc.

Bandura, A. (2004). *Observational Learning, in J. H. Byrne (Ed.) Learning and Memory*. (2nd ed.). New York: Macmillan Reference USA.

Barrett, S., & Stone, P. (2011). Ad Hoc Teamwork Modeled with Multi-armed Bandits : An Extension to Discounted Infinite Rewards Categories and Subject Descriptors. *Teacher*, (May).

Barrett, S., Stone, P., & Kraus, S. (2011). Empirical evaluation of ad hoc teamwork in the pursuit domain. In *AAMAS* (pp. 567–574). Retrieved from http://dl.acm.org/citation.cfm?id=2031678.2031698

Barrett, S., Stone, P., Kraus, S., & Rosenfeld, A. (2012). Learning teammate models for ad hoc teamwork. *AAMAS Adaptive Learning Agents (ALA) Workshop*, 57–63. Retrieved from http://u.cs.biu.ac.il/~sarit/data/articles/ala2012.pdf

Berry, P., Peintner, B., Conley, K., Gervasio, M., Uribe, T., & Yorke-Smith, N. (2006). Deploying a personalized time management agent. *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*. ACM.

Bouron, T., Ferber, J., & Samuel, F. (1990). MAGES: A Multiagent Testbed for Heterogeneous Agents. *Decentralized Artificial Intelligence 2*, 221–239.

Caillou, P., Aknine, S., & Pinson, S. (2002). How to Form and Restructure Multi-agent Coalitions. *National Conference on Artificial Intelligence (AAAI 02) Workshop on Coalition Formation*. Edmonton, Canada: AAAI Press.

Chalupsky, H., Gil, A., Knoblock, C. A., Lerman, K., Oh, J., Pynadath, D. V, … Tambe, M. (2002). Electric Elves: Agent Technology for Supporting Human Organizations. *AI MAGAZINE*, *23*, 11–24. https://doi.org/10.1.1.111.5145

Chen, B., Chen, X., Timsina, A., & Soh, L. (2015). Considering Agent and Task Openness in Ad Hoc Team Formation. *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, (Aamas), 1861–1862.

Chen, B., Eck, A., & Soh, L. (2016). Collaborative Human Task Assignment for Open Systems (Extended Abstract). *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*, (Aamas), 1441–1442.

Fullam, K. K., Klos, T. B., Muller, G., Sabater, J., Schlosser, A., Topol, Z., … Voss, M. (2005). A specification of the Agent Reputation and Trust (ART) testbed: experimentation and competition for trust in agent societies. *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, 512–518. https://doi.org/10.1145/1082473.1082551

Heidig, S., & Clarebout, G. (2011). Do pedagogical agents make a difference to student motivation and learning? *Educational Research Review*. https://doi.org/10.1016/j.edurev.2010.07.004

Henderson, B. D. (1984). The application and misapplication of the experience curve. *Journal of Business Strategy*, *4*(3), 3–9.

Hewitt, C. (1986). Offices are open systems. *ACM Transactions on Information Systems (TOIS)*, *4*(3), 271–287. Retrieved from http://dl.acm.org/citation.cfm?id=214432

Huynh, T. D., Jennings, N. R., & Shadbolt, N. R. (2006). An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, *13*(2), 119–154. https://doi.org/10.1007/s10458-005-6825-4

Jamroga, W., Męski, A., & Szreter, M. (2013). Modularity and Openness in Modeling Multi-Agent Systems. *Electronic Proceedings in Theoretical Computer Science*, *119*, 224–239. https://doi.org/10.4204/EPTCS.119.19

Jennings, N. R., Moreau, L., Nicholson, D., Ramchurn, S., Roberts, S., Rodden, T., & Rogers, A. (2014). Human-agent collectives. *Communications of the ACM*, *57*(12), 80–88. https://doi.org/10.1145/2629559

Jumadinova, J., Dasgupta, P., & Soh, L. K. (2014). Strategic capability-learning for improved multiagent collaboration in Ad Hoc environments. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *44*, 1003–1014. https://doi.org/10.1109/TSMC.2013.2285527

Kamar, E., Gal, Y., & Grosz, B. J. (2013). Modeling information exchange opportunities for effective human-computer teamwork. *Artificial Intelligence*, *195*, 528–550. https://doi.org/10.1016/j.artint.2012.11.007

Khandaker, N., & Soh, L. K. (2007). Formation and Scaffolding Human Coalitions in I-MINDS—A Computer-Supported Collaborative Learning Environment. *AAMAS Agent-Based Systems for Human Learning & Entertainment Workshop*, 64–75.

Khandaker, N., Soh, L. K., Miller, L. D., Eck, A., & Jiang, H. (2011). Lessons learned

from comprehensive deployments of multiagent CSCL applications I-MINDS and ClassroomWiki. *IEEE Transactions on Learning Technologies*, *4*(1), 47–58. https://doi.org/10.1109/TLT.2010.28

Leibowitz, N., Baum, B., Enden, G., & Karniel, A. (2010). The exponential learning equation as a function of successful trials results in sigmoid performance. *Journal of Mathematical Psychology*, *54*(3), 338–340. https://doi.org/10.1016/j.jmp.2010.01.006

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM*, *37*(7), 31–40. https://doi.org/10.1145/176789.176792

Manson, N. C., & O'Neill, O. (2007). *Rethinking informed consent in bioethics*. Cambridge University Press.

Marcolino, L. S., Jiang, A. X., & Tambe, M. (2013). Multi-agent team formation: Diversity beats strength? *IJCAI International Joint Conference on Artificial Intelligence*, 279–285.

Myers, K., Berry, P., Blythe, J., & Conley, K. (2007). An intelligent personal assistant for task and time management. *AI Magazine*, *28*(2), 47–62. https://doi.org/10.1609/aimag.v28i2.2039

Newell, A., & Rosenbloom, P. (1993). Mechanisms of skill acquisition and the law of practice. *The Soar Papers*, (1), 81–135. Retrieved from http://dl.acm.org/citation.cfm?id=162586

Pinyol, I., & Sabater-Mir, J. (2013). Computational trust and reputation models for open multi-agent systems: A review. *Artificial Intelligence Review*, *40*(1), 1–25. https://doi.org/10.1007/s10462-011-9277-z

Roediger, H. L., & Smith, M. a. (2012). The "pure-study" learning curve: The learning curve without cumulative testing. *Memory & Cognition*, *40*(7), 989–1002. https://doi.org/10.3758/s13421-012-0213-5

Russell, S., & Norving, P. (1995). *Artificial Intelligence: a Modem Approach*. Englewood Cliffs, NJ: Prentice-Hall.

Shehory, O. (2001). Software architecture attributes of multi-agent systems. In *Agent-Oriented Software Engineering* (pp. 77–90). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/3-540-44564-1_5

Shehory, O., & Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, *101*(1–2), 165–200. https://doi.org/10.1016/S0004-3702(98)00045-9

Shell, D. F., Brooks, D. W., Trainin, G., Wilson, K. M., Kauffman, D. F., & Herr, L. M. (2010). *The unified learning model: How motivational, cognitive, and neurobiological sciences inform best teaching practices*. Springer Netherlands. https://doi.org/10.1007/978-90-481-3215-7

Shoham, Y., & Leyton-Brown, K. (2008). *Multiagent Systems: Algorithmic, Game-*

Theoretic, and Logical Foundations. Cambridge University Press.

Sklar, E., & Richards, D. (2006). The use of agents in human learning systems. *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 767–774. https://doi.org/10.1145/1160633.1160768

Smith, R. G. (1980). The contract net protocol: high level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, *29*(12), 1104–1113.

Soh, L. K., & Tsatsoulis, C. (2002). Satisficing coalition formation among agents. *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3*. ACM. https://doi.org/10.1145/545068.545071

Stone, P., Gan, R., & Kraus, S. (2010). To teach or not to teach? Decision making under uncertainty in ad hoc teams. *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, 117–124.

Stone, P., Kaminka, G. A., & Rosenschein, J. S. (2010a). Leading a best-response teammate in an ad hoc team. *Lecture Notes in Business Information Processing*, *59 LNBIP*(May), 132–146. https://doi.org/10.1007/978-3-642-15117-0_10

Stone, P., Kaminka, G. a, & Rosenschein, J. S. (2010b). Ad Hoc Autonomous Agent Teams : Collaboration without Pre-Coordination. *Twenty-Fourth AAAI Conference on Artificial Intelligence*, (July), 1504–1509.

Tambe, M. (2008). Electric Elves: What Went Wrong and Why. *AI Magazine*, *29*(2), 23–31. https://doi.org/10.1609/aimag.v29i2.2123

Vassileva, J., McCalla, G. I., & Greer, J. E. (2015). From Small Seeds Grow Fruitful Trees: How the PHelpS Peer Help System Stimulated a Diverse and Innovative Research Agenda over 15 Years. *International Journal of Artificial Intelligence in Education*, 1–17. https://doi.org/10.1007/s40593-015-0073-9

Vygotsky, L. S. (1978). *Mind in Society. Memory*. Boston, MA: Harvard University Press.

Wifall, T., McMurray, B., & Hazeltine, E. (2014). Perceptual Similarity Affects the Learning Curve (but Not Necessarily Learning). *Journal of Experimental Psychology: General*, *143*(1), 312–331. https://doi.org/10.1037/a0030865

Wooldridge, M. (2009). *An Introduction to Multiagent Systems*. Chichester, U.K: Wiley.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, *10*(2), 115–152. https://doi.org/10.1017/S0269888900008122

Wu, F., Zilberstein, S., & Chen, X. (2011). Online planning for ad hoc autonomous agent teams. *IJCAI International Joint Conference on Artificial Intelligence*, 439–445. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-081

Ying, C. C. (1967). LEARNING BY DOING-AN ADAPTIVE APPROACH TO MULTIPERIOD DECISIONS. *Operations Research*, *15*(5), 797–812. Retrieved

from
http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=4468934&site=ehost-live&scope=site

Zhang, Y., & Parker, L. E. (2012). Task allocation with executable coalitions in multirobot tasks. In *Proceedings - IEEE International Conference on Robotics and Automation* (pp. 3307–3314). https://doi.org/10.1109/ICRA.2012.6224910