Strategic Capability-Learning for Improved Multi-Agent Collaboration in Ad-hoc Environments

Janyl Jumadinova, Prithviraj Dasgupta, and Leen-Kiat Soh

***Abstract*—We consider the problem of distributed collaboration among multiple agents in an ad-hoc setting. We have analyzed this problem within a multi-agent task execution scenario where every task requires collaboration among multiple agents to get completed. Tasks are also ad-hoc in the sense that they appear dynamically, and, require different sets of expertise or capabilities from agents for completion. We model collaboration within this framework as a decision making problem where agents have to determine what capabilities to learn and which agents to learn them from so that they can form teams which have the capabilities required to perform the current tasks satisfactorily. Our proposed technique refers to principles from human learning theory to enable an agent to strategically select appropriate capabilities to learn from other agents. We also use two 'openness' parameters to model the dynamic nature of tasks and agents in the environment. Experimental results within the Repast agent simulator show that by using the appropriate learning strategy, the overall utility of the agents improves considerably. The performance of the agents and their utilities are also dependent on the repetitiveness of tasks and re-encounter with agents within the environment. Our results also show that the agents that are able to learn more capabilities from another expert agent outperform the agents that learn only one capability at a time from many agents, and, agents that use an intelligent utility maximizing strategy to choose which capabilities to learn outperform the agents that randomly make the learning decision.**

***Index Terms*—Multi-agent systems, learning, collaborative work, ad hoc networks.**

I. Introduction

Collaboration among a set of autonomous agents is an important research topic within multi-agent systems with applications in several domains such as robotic search and rescue, robotic foraging, robotic pursuit evasion, etc. [1]. In real-life, most of the scenarios in which humans collaborate are unstructured and ad-hoc. The scenarios do not specify the desired collaborative behavior and skills required by the humans beforehand, but usually require the humans to adapt, evolve and acquire their behaviors and skills as the collaboration proceeds to meet the desired goal of the collaboration process. However, in many autonomous agent-based systems, the agents exhibit a specific, pre-determined and perhaps inflexible behavior, which is usually controlled by a coordination mechanism. While using such preset coordination mechanisms provides advantages such as allowing each agent in the system to make utility-maximizing decisions and to measure the progress of the agents' actions towards achieving the overall goal of the system, it limits the flexibility or adaptation of the behaviors of the autonomous agents. Unfortunately, such inflexible behavior by the agents limits their suitability as human-aids or human-substitutes in many real-life collaboration scenarios. Therefore, it makes sense to investigate techniques for building agent-based autonomous systems that will be capable of handling diverse, dynamic and ad-hoc collaboration situations in an efficient manner.

An attractive model for enabling collaboration among ad-hoc teams [2] is the teacher-learner framework. We surmise that when human agents work together, it is inevitable that they learn from each other, and even on occasions teach each other. Thus, it would be prudent to factor this into the team formation process. Indeed, real-world human team formation has exhibited this phenomenon. For example, parents, teachers, and employers have formed a team with members of lesser joint capabilities-as long as they can afford to not solve the task at hand optimally-so that team members have a chance to learn or teach. Towards leveraging the above premise, we further submit three important considerations:

- First, in ad hoc team formation where team members had no prior knowledge about each other, it could be advantageous to consider the impact of learning so that agents get to benefit more than just from accomplishing a task. This type of learning

Manuscript received XXX; revised YYY.

J. Jumadinova is with the Department of Computer Science, Allegheny College, e-mail: jjumadinova@allegheny.edu.

P. Dasgupta is with the Department of Computer Science, University of Nebraska - Omaha, e-mail: pdasgupta@unomaha.edu.

L.-K. Soh is with the Department of Computer Science and Engineering, University of Nebraska - Lincoln, e-mail: lksoh@cse.unl.edu.

and teaching or knowledge transfer is spontaneous, ad hoc in many ways as well, and reflects well what real-world ad hoc teams do.

- Second, it is appropriate and even strategic to consider agent openness and task openness in this context when considering this type of learning. For example, an agent who is lacking in a particular capability may opt to join a team with a good opportunity to learn from another agent about that particular capability even when the direct rewards from accomplishing the team's task might not be sufficiently enticing. Thus, if the degree of agent openness is high in the environment-i.e., it is unlikely or very uncertain that the agent with the desired capability might be around in the environment-then it would be advisable, for example, for the agent to lean towards joining a team now to work with that particular agent. In addition, if the degree of task openness is high in the environment-such that a task that requires a uniquely particular capability is unlikely or very uncertain to appear in the environment again-then the agent might afford to have a relatively low motivation to want to acquire that capability which would be in a way obsolete or unlikely to be put into good use.
- Third, the area of education psychology, especially in student or human learning and teaching, has described different types of knowledge transfer when humans are involved in teamwork. These learning and teaching processes, implicit or explicit, direct or indirect, come with different cost in terms of time and effort (involvement) and cognitive loads of the learning and teaching parties. To maintain our focus on the impact and benefits of learning and teaching on the decisions and dynamics involved in ad-hoc team formation, we have not implemented the actual learning process in our model. Instead, we have considered the learning process as a separate subsystem or a 'blackbox' and used mapped, discretized costs and the effectiveness of different types of learning in our model. The factors enabling or facilitating each type of learning or teaching when estimating their costs and effectiveness is a planned next phase of our research.

Our work in this paper uses a teacher-learner approach for ad-hoc collaboration in a multi-agent task execution scenario. In our scenario, tasks can be completed by a group of agents that have the necessary expertise or capabilities for performing the task. Each agent receives a certain utility when it performs its portion of a task, along with a much larger utility if the task is completed satisfactorily by the group that is selected to perform the task. Agents are utility maximizers. Consequently, each agent tries to improve its chances of getting selected to perform a task by acquiring the capabilities that align with the capabilities that are required to perform current tasks in the environment. This problem is not straightforward as tasks are introduced dynamically in the environment and agents that possess useful key capabilities to perform tasks might leave the environment dynamically. Our proposed teacher-learner framework addresses this problem by developing decision-making strategies that can be used by an agent to strategically determine what the 'best' capabilities to learn from other agents are, and which agents are the 'best' teachers to learn those capabilities from. Our proposed learning framework is based on principles from human learning theory [3], [4]. Each agent also employs two parameters called *agent openness* and *task openness* to model the dynamic nature of the agents and tasks in the environment.

We have simulated the operation of our multi-agent system for ad-hoc collaboration on the Repast agent simulator. Our results show that the agents that use our teacher-learner framework to decide what capabilities to learn from other agents get more utility than the agents that select capabilities to learn randomly or with a fixed probability distribution. While we also find that the agents in more stable environments are able to learn and complete tasks more effectively than the agents in more dynamic environments, in addition we also observe an unexpected behavior where agents that try to learn all capabilities (not necessarily optimally) outperform other agents that optimize or select one capability to learn.

II. Related Work

Multi-agent collaboration has been an important research topic in multi-agent systems [5]. However, in [1] the authors observe that not much of this research has focused on addressing the ad-hoc nature of the environment where agents might not know the coordination protocol used by each other, might not share the same world model, and even might not be able to communicate directly with each other. Recently different facets of the ad-hoc team collaboration problem have been studied. Stone and Kraus [6] considered the problem of ad hoc team collaboration by two agents, agent A, the teacher, and agent B in the k-armed bandit problem. They analyzed the task of agent A (which actions it should choose) in order to maximize team's utility in a setting where a teammate has limited action capabilities and a fixed and known behavior. In [7], the authors consider the problem of a single ad-hoc team player leading other team-mates that use a best response strategy to select their actions,

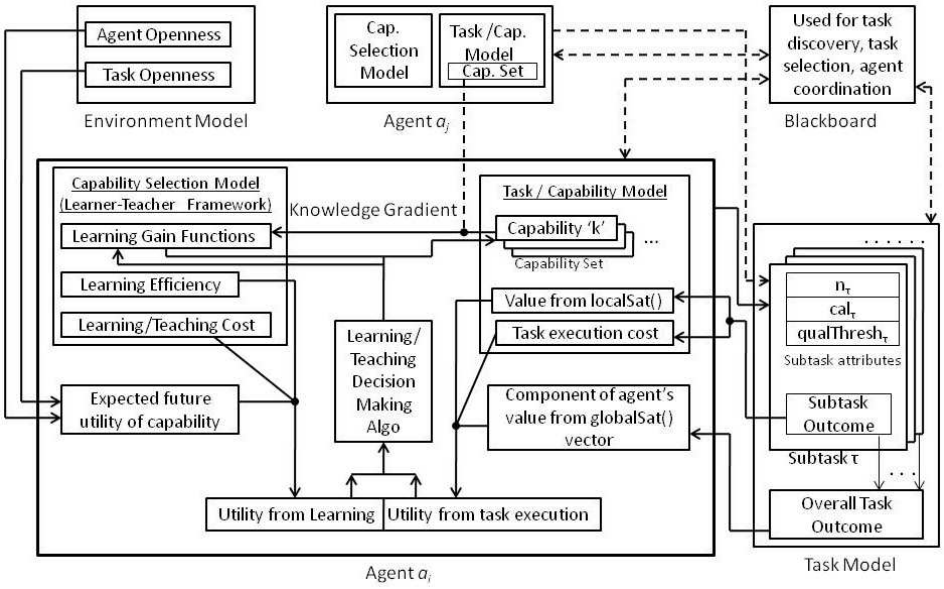

Fig. 1. Task Execution and capability learning components showing agents a_i and a_j that work together on subtask τ within the ad-hoc multi-agent collaboration model.

towards the optimal (cost-minimizing) joint action. They use a graphical modeling approach to solve this type of ad hoc game initially with three players and then with N-players. Barrett *et al.* [8] propose a framework for analyzing ad hoc team problems by describing a set of dimensions. They analyze several traditional games in pursuit domain and a few simple ad hoc games using three dimensions: team knowledge, environmental knowledge, and member reactivity.

Albrecht *et al.* [9] empirically evaluate five multi-agent learning algorithms in ad hoc situations with 2 players and 3 players. They compare these algorithms in five aspects: convergence rate, final expected payoff, social welfare, social fairness, and rate of solution types. In [10], the authors consider the problem of determining the roles for a set of "inside" or controllable agents, so that the marginal utility of an ad-hoc team comprising both controllable and outside or uncontrollable agents is maximized. Agent capabilities for each role are assumed to be unchanged. In contrast, in our work, the capabilities of agents can be dynamically updated based on expected future interactions with other agents and the type of future tasks and agents can improve their utilities by acquiring sought-after capabilities. In [11], the authors consider teams of agents working together where each agent acquires capabilities in proportion to the utility derived by performing tasks with other agents. Like [10], their problem is formulated as selecting agent roles by determining the optimum role-mapping policy. In contrast, in our work, agents can acquire capabilities or skills through learning/teaching, outside of doing tasks so that they can improve their utilities by using those capabilities later on.

Some researchers have developed personal assistant agents to support users working within collaborative design environments (CDE). Wua *et al.* [12] use personal assistant agents to effectively work with the corresponding users to achieve their goals in a collaborative environment, while Myers *et al.* [13] develop an intelligent personal assistant to help a busy worker in managing time commitments and performing tasks. Such work on personal assistant agents concentrates more on a user interaction and enhancing the user experience.

Another research area related to learning in ad-hoc environments is e-Learning, which encompasses a wide spectrum of applications and processes such as Web-based education, virtual classrooms and digital collaboration. In [14], the authors describe a student modeling server, a part of an architecture that allows independent teams to develop user-adaptive components that could interact in parallel with the same user. This architecture can also integrate collected information, resulting in better adaptations for the user. Baylari *et al.* [15] propose a personalized multi-agent e-learning system based on item response theory and an artificial neural network. Their system presents adaptive tests and personalized recommendations, where the agents add adaptivity and interactivity to the learning environment and act as a human instructor to guide the learners in a friendly and personalized teaching environment.

In contrast, the focus of our work in this paper is not focused on collaborative learning nor human learning, though we agree that these are critical steps that are of interests in our next phases of research. Instead, our work is focused on the following premise. In ad hoc team formation (or general team formation, for that matter), when deciding which team to join or form, an agent often bases its decision on the expected utility of the rewards from accomplishing a task or subtask leading to satisfaction of a joint goal(s). What we propose here is to add another level of "rewards" to this decision making process-one that involves the indirect, or implicit rewards that would come in the future from an agent working with the others in a team. That is, the indirect rewards stem from the agent learning from the others and acquiring better capabilities from working in a team, or from an agent teaching other agents how to solve a task better.

III. Multi-agent Ad-hoc Collaboration Model

We motivate our ad-hoc collaboration scenario within a search-and-rescue domain which involves groups of people, with different expertise and capabilities, working together to perform search-and-rescue related tasks. In a search-and-rescue domain, there is usually no central coordinator to assign persons or teams to tasks. Consequently, the individuals and groups have to determine by themselves what tasks they can perform, and, collaborate within their group as well as with other groups involved in the search-and-rescue operations to perform the tasks. Such a scenario provides an illustrative and appropriate setting of ad-hoc collaboration. Consider a search-and-rescue mission following a big disaster which lasts over a period of several days or weeks. Each member of the search-and-rescue team receives a one-time reward based on the quality of that person's tasks performed in the search-and-rescue mission. Additionally, the entire team receives a large reward at the end of the mission, if it was successful. Events such as natural disasters that require search-and-rescue efforts recur frequently (e.g., once or twice a year). So, the people that took part in a search-and-rescue team may be called on again or another search-and-rescue effort of a similar or different nature (type). Each member of the search-and-rescue team has different skills for performing search-and-rescue related tasks. Every member is also rational he wants to perform his task as well as possible so that he can get the maximum reward for the current search-and-rescue mission and at the same time improve the chances of him being invited to be a member of another -search-and-rescue team in the future. Additionally, the members of the search-and-rescue team are far-sighted and they want to participate in other search-and-rescue missions in the future to increase their rewards. To be able to do this, the members want to learn skills or capabilities related to the current search-and-rescue tasks from the current members, especially from the ones who have a good experience in such operations. Learning capabilities incurs a learning cost. The members also need to keep in mind that search-and-rescue missions can be of different types. Therefore, besides learning capabilities required for the current search-and-rescue tasks, they should also consider learning capabilities that might be useful in other types of search-and-rescue missions, from other members in the team. Finally, some members leave the team from time to time, so it might be beneficial to learn from an expert member, in case he has already left the team at a future time when his skill becomes necessary. We assume that each member of a search-and-rescue team has an autonomous software agent associated with him that performs decision making calculations and relays the result to the human. We also assume that the human follows the decision made by the software agentin a way, the software agents in our environment represent their human counterparts. The ad-hoc collaboration problem that we address in this paper is the learning problem facing the members of the search-and-rescue team - what aspects of different types of search-and-rescue operations should they learn, who should they learn from and when should they learn; so that they can ultimately increase their rewards in the long-term and receive satisfaction to offset the learning expenditure. Our framework for multi-agent ad-hoc collaboration consists of two main components for each agent, a task execution model and a capability learning/teaching model. The functionality of each of these models is explained below:

A. Task and Capability Model

Let $\overline{\mathcal{T}}$ be a set of tasks and A be a set of agents that can perform those tasks. Each task can be decomposed into a set of subtasks and agents complete the task by coordinating with each other to perform the subtasks. Each task $\mathcal{T} \in \overline{\mathcal{T}}$ has an associated task type $\mathcal{T}_{type}$ that is determined by the set of subtasks comprising the task. $\overline{\mathcal{T}}_{\boldsymbol{type}}$ is the set of all task types. For the sake of legibility, we refer to a task and associated set of subtasks it is decomposed into with the same notation $\mathcal{T}$. Each agent has a set of resources and can execute actions using those resources to perform tasks. We have abstracted the set of actions and resources of an agent as a set of task-related capabilities of the agent. Let Cap

denote the set of capabilities or skills that agents in A possess and $cap_i \subseteq Cap$ denote the capabilities available with agent a_i. The quality of a capability k of agent a_i, $cap_{i,k}$, is associated with numerical quality value, $qual(cap_{i,k}) \in [0, 1]$. An agent can improve the quality of its existing capabilities through learning from its own actions or from other agents, while not using a capability consistently results in a gradual deterioration of the capability's quality. The dynamic update of capabilities through learning from other agents in this manner allows the agents to dynamically form teams that specialize in capabilities that are most useful to perform current task requirements in the environment.

B. Task Discovery and Agent Selection

Let $A_{\mathcal{T}} \subseteq A$ denote the set of agents that have the capabilities to perform at least one of the subtasks of $\mathcal{T}$. The interaction and coordination between multiple agents to allocate and perform the subtasks within a task is implemented through a blackboard-based publish-subscribe system [16]. Using an open interaction architecture like a blackboard, precludes the need to explicitly provide specific coordination and communication protocols between agents, as mentioned in [1]. A task is introduced into the environment when an agent, $a_{disc} \in A$ discovers it. The task discovery protocol is assumed to be domain specific. The agent a_{disc} decomposes the task $\mathcal{T}$ into a set of disjoint subtasks such that each subtask $\tau \in \mathcal{T}$ requires exactly one capability from the capability set Cap from one or more agents. All the subtasks are published on the blackboard by a_{disc}. Each subtask $\tau_j \in \mathcal{T}$ is associated with three parameters, the capability cap_{τ_j} required for it, the minimum number of agents, n_{τ_j} with that capability required to perform it, and, the minimum quality threshold $qualThresh_{\tau_j}$ that gives the minimum quality of the capability that each agent performing it should have. $qualThresh_{\tau_j}$ is assumed to be a parameter that is specific to the scenario at hand. It is determined from the scenario's operational conditions and provided by a human domain expert. In general, in a scenario where optimization of task quality is more important than, say, mere satisfaction of task completion, the threshold should be set high. An example of this scenario is making travel arrangements to minimize cost while maintaining quality of serviceflight, accommodation, etc. for a trip in a short time frame. On the other hand, in a scenario where satisfaction of a task completion is more viable than quality optimization, say, due to stringent resource constraints such as lack of qualified personnel in a disaster response situation, then the threshold can be set lower. From another perspective, if the system consists of mostly experts, then setting a high threshold would ensure that lesser agents are unlikely to be called to complete tasks, and vice versa. Note also that effects of teaching and learning are more likely to be felt when a team has agents of different levels of expertise. So, in a way, to encourage teaching and learning, it might be strategic to choose a low threshold to increase the chance of having a mixture of agents in a team. In our experiments to be discussed later in Section V, we have experimented with various quality threshold levels for different task types ranging from 0.2 (low threshold) to 0.7 (high threshold), to have tasks with diverse quality threshold requirements.

Each agent a_i uses a distributed auction mechanism given in Algorithm 1 to select subtasks [17]. In the auction-based task assignment mechanism, each subtask τ_j is associated with a virtual price p_j that is initialized to 0. The algorithm proceeds in rounds. In each round, an agent a_i that does not have a subtask assigned to it selects two subtasks, τ_{j_1} and τ_{j_2}, which it is capable of performing, and, for which a_i has the largest and the second largest difference between its quality of the capability and the current price of the subtask, respectively. Agent a_i then submits a bid b_i given by the difference between these highest and second highest values, plus a small, real-valued, bid increment denoted by δ. The bid-increment ensures that the algorithm terminates if multiple agents have the same quality of capability for doing a certain subtask. If the bid b_i is non-zero, agent a_i is assigned subtask τ_{j_1}. If the number of agents assigned to perform τ_{j_1} exceeds $n_{\tau_{j_1}}$, then the agent with the lowest bid is removed. The virtual price corresponding to τ_{j_1} is also updated by b_i. The algorithm terminates when every agent has an assigned subtask. On termination, the algorithm guarantees that the n_{τ_j} agents that have the highest quality of capabilities for subtask τ_j are allocated to perform it. Agents that are allocated to perform the same subtask coordinate with each other using some coordination protocol [16] to perform their actions on the task. Because task allocation and agent coordination are not central to the collaboration problem discussed here, we do not detail it further in this paper. We assume that different existing techniques [16] could be used for performing these activities depending on the application domain.

C. Agent Utility Model

Let $\tau_i \subseteq \mathcal{T}$ denote the set of subtasks of $\mathcal{T}$ that agent a_i is performing and $\tau_{i,j}$ denote its j-th subtask. When a subtask $\tau_{i,j}$ is completed, the agent a_i that performed it using capability k at quality level $qual(cap_{i,k})$, associates a numerical *satisfaction* value to denote how

selectSubTasksWithAuction($Blackboard\ b$)
Set S; // set with assignment pairs (a_i, τ_j)
foreach *subtask* τ_j *of* $\mathcal{T}$ *in* b **do**
  // Determine subset of agents that have capability to do subtask τ_j
  $A'_j \leftarrow \{a_i | (cap_{\tau_j} = cap_{i,k}) \wedge (qual(cap_{i,k}) > qualThresh_{\tau_j})\}$
end
$S \leftarrow \{\emptyset\}$;
foreach $\tau_j \in \mathcal{T}$ **do**
  $p_j \leftarrow 0$;
end
repeat
  Let $a_i \in A'_j$ be an unassigned agent that is capable of doing τ_j
  //Find subtasks τ_{j_1} and τ_{j_2} which offer highest and second highest difference between quality and current price
  $j_1 \leftarrow \arg\max_{k|(i \in A'_{j_1})}(qual(cap_{i,k}) - p_k)$
  $j_2 \leftarrow \arg\max_{k|(i \in A'_{j_2}, k \neq j_1)}(qual(cap_{i,k}) - p_k)$
  //Compute a_i's bid increment for τ_{j_1}
  $b_i \leftarrow$
  $(qual(cap_{i,j_1}) - p_{j_1}) - (qual(cap_{i,j_2}) - p_{j_2}) + \delta$
  if $b_i > 0$ **then**
    //Assignment of agent to subtask
    Add the pair (a_i, τ_{j_1}) to the assignment S
    if *number of assignments of* τ_{j_1} *in* $S > n_{\tau_{j_1}}$ **then**
      remove (a'_i, τ_{j_1}) from S where a'_i had minimum bid for τ_{j_1}
    end
    $p_{j_1} \leftarrow p_{j_1} + b_i$
  end
until *every agent* a_i *has an assigned subtask*;

Algorithm 1: Auction algorithm used by agents to select subtasks

satisfactorily the subtask was completed. This value is given by a local satisfaction function $localSat : \mathcal{T} \times [0,1]^{n_\tau} \rightarrow [0,1]$. $localSat(\tau_{i,j}, qual(cap_{i,k})) = 1(0)$ means that the subtask was completed most satisfactorily (unsatisfactorily). When all of the subtasks of task $\mathcal{T}$ are completed, a global satisfaction function, $globalSat : \times_{\tau \in \mathcal{T}} \ \tau \times [0,1]^{n_\tau} \rightarrow [0,1]$, is used to denote how satisfactorily the task was completed. It is calculated and provided by an external "reviewer" entity that assesses the outcome of the overall task based on the outcome of the subtasks. The external entity could be a human or a software agent. It is assumed to have sufficient domain expertise to assess the quality of the task's outcome and assign a reward value for the performers (agents) of the task. As before, $globalSat(\cdot) = 1$ denotes the task was completed most satisfactorily. The value derived by an agent $a_i \in A_\mathcal{T}$ for performing task $\mathcal{T}$ is given as a function of the local and global satisfaction levels, i.e., $V_i(\mathcal{T}) = \sum_{\tau \in \mathcal{T}} V_i(\tau, localSat(\cdot)) + V_i(\mathcal{T}, globalSat(\cdot))$. That is, the value that the agent a_i receives at the completion of the whole task $\mathcal{T}$ consists of the value that the agent a_i gets for performing a subtask τ of the task $\mathcal{T}$ and the value the agent a_i gets for participating in completing the whole task $\mathcal{T}$, after all the subtasks associated with task $\mathcal{T}$ are finished. Local satisfaction value is determined as $V_i(\tau, localSat(\cdot)) \sim \mathcal{N}(qual(cap_{i,k}, 1)$ and the global satisfaction function is set as: $V_i(\mathcal{T}, globalSat(\cdot)) \sim \mathcal{N}(\sum_{j=1}^{|\tau_i|} localSat(\cdot), 1)$.

Each agent $a_i \in A_\mathcal{T}$ incurs an expenditure or cost denoted by $c_{\tau_{i,j}}$ for the actions it does to perform subtask $\tau_{i,j}$ and the total cost to agent a_i for performing task $\mathcal{T}$ is given by the sum of its costs for each subtask in $\mathcal{T}$ that it performed, $C_i(\mathcal{T}) = \sum_{\tau_{i,j} \in \mathcal{T}} c_{\tau_{i,j}}$. We have assumed that each agents utility is calculated independently based on its performance of the each subtask it performs. In addition, the final outcome of the task also contributes a portion of the utility received by the agent through the value of global satisfaction of the task that is calculated by an external entity for each agent. Specifically, each agent a_i has a utility for task $\mathcal{T}$ given by the following utility function:

$$U_i(\mathcal{T}) = V_i(\mathcal{T}) - C_i(\mathcal{T}). \quad (1)$$

Thus, the utility agent a_i receives after completing task $\mathcal{T}$ is the sum of the values it receives for performing subtasks related to $\mathcal{T}$ and the value it receives after $\mathcal{T}$ is completed, minus the costs the agent a_i incurs for performing subtasks of $\mathcal{T}$.

Figure 1 shows the task and capability models that is used by an agent in our proposed framework.

D. Capability Learning Model

When agents especially humans collaborate, it is likely that agents learn from their collaborative experiences, and these learning episodes lead to changes in their capabilities and subsequent decision making. Moreover, in an ad-hoc environment, tasks appear dynamically and the distribution of the arrival of tasks is not known *a priori* to the agents. Consequently, the set of capabilities and corresponding qualities that are required to perform tasks can vary dynamically. In a way, an agent should acquire or improve capabilities that are in high-demand, so that it can participate in performing tasks requiring those capabilities as experts and improve its utility. However, due to the openness in the environment, an

agent needs to decide what capabilities to learn, when to learn them and from whom to learn capabilities, without pre-coordination.

To this end, we have used the learning-teaching model between humans as a basis for the corresponding model used in our ad-hoc collaboration scenario. Human learning is characterized by different learning types. In [18], the authors describe six different learning types. Each learning type is associated with a learning cost and a learning efficiency that denotes how effective the learning type is. Table I shows the different learning types and the ranking of corresponding costs and efficiencies. For our model, we consider the four learning types that have distinct costs and efficiencies in the table. We denote L_{type} as the set of learning types and $clearn(lt)$ and $elearn(lt)$ as the cost and efficiency of learning type $lt \in L_{type}$ respectively.

Rank of Cost and Eff.	Learning Type
4	Teaching/Guiding
4	Being taught
4	Apprenticeship
3	Learning by Discussion
2	Learning by Practice
1	Learning by Observation

TABLE I
DIFFERENT LEARNING TYPES AND RANKING OF THEIR COSTS AND EFFICIENCIES. HIGHER RANK MEANS HIGHER COST AND EFFICIENCY.

The first step for an agent a_i that intends to acquire useful capabilities is to determine the set of capabilities that give it the highest learning gain. In human learning scenarios, when one human learns from another the amount of information transferred from the teacher to the learner is proportional to the knowledge gradient between them. Following this approach, we model the learning gain between two agents a_i and a_j for capability k to be proportional to the capability difference between them given by $qual(cap_{i,k}) - qual(cap_{j,k})$ (in the rest of the paper we assume $a_j \in A_{\mathcal{T}} \setminus \{a_i\}$). Designing an appropriate function to quantify the learning gain while modeling human learning requires some insight. Vygotsky's zone of proximal development (ZPD) theory [4] 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 have designed the learning gain function between agents a_i and a_j for capability k using learning type $lt \in L_{type}$ as the following function:

$$Gain(lt, a_i, a_j, k) = \begin{cases} \frac{\eta}{qual(cap_{i,k})+\epsilon}, & \text{if } lt = \text{self-learning} \\ \frac{c^2-[(qual(cap_{i,k})-qual(cap_{j,k}))-c]^2}{qual(cap_{i,k})+\epsilon}, & \text{otherwise} \end{cases}$$

where η is a constant denoting the increment in knowledge from self-learning, ϵ is a small number in case $qual(cap_{i,k})$ is zero and c is a constant to cap the learning gain. For learning types other than self-learning, the learning gain between agents a_i and a_j is maximum when their qualities for capability k have a difference c. When their quality values are very close to each other or very far apart, the learning gain decreases towards zero.

Agent and Task Openness. The learning gain calculated by an agent identifies its potential benefit from learning different capabilities from different agents. However, it does not address the question whether the capability learned by the agent will be useful for it to perform tasks in the future and improve its task-execution utility (given in Equation 1). In an ad-hoc environment, if an agent is deciding on whether to learn about the capabilities required for a particular subtask, it should also consider the likelihood that it will encounter the same subtask again in the future. Likewise, if an agent a_i has to make a decision on whether to learn capability k_1 or capability k_2 from agents a_{j1} and a_{j2} respectively, it should include the likelihood that it will encounter agent a_{j1} or a_{j2} again in the future and have a chance to learn the capability from it later on. Combining the above factors, the expected utility of learning about a capability should weigh the knowledge gain from learning by the likelihood of working with an agent again and of encountering a task requiring that capability again in the future. To address this question each agent uses two parameters called the *task openness* and *agent openness* to respectively model the type of tasks that can be expected in the future, and, the expected availability of agents from whom the capabilities required to perform the expected future tasks can be learned. Let $p_i(a_j \mid A_{\mathcal{T}})$ denote the likelihood of a_i working with a_j in ad hoc team $A_{\mathcal{T}}$. Agent openness is defined as $1 - p_i(a_j \mid A_{\mathcal{T}})$. Let $p_i(cap_{i,k} \mid \mathcal{T})$ denote the likelihood of agent a_i using capability k to solve a subtask in task $\mathcal{T}$ and $p(\mathcal{T})$ denote the likelihood of task $\mathcal{T}$ appearing again in the future. Task openness is defined as the product of these two probability values, $p_i(cap_{i,k} \mid \mathcal{T}) \cdot p(\mathcal{T})$. If the agent openness is high, new agents appear more often and $p_i(a_j \mid A_{\mathcal{T}})$ is low. If task openness is high, new types of tasks appear more often and capability k of agent i, $cap_{i,k}$, might not be needed again. In that case,

$p(cap_{i,k} \mid \mathcal{T})$ will be low. And vice versa.

E. Capability Learning Strategies

The expected utility of an agent a_i learning from a_j about a capability k using the learning type lt is given by:

$$\begin{aligned} U_i(lt, a_j, k) = & \\ & elearn(lt) \cdot Gain(lt, a_i, a_j, k) \cdot p_i(a_j \mid A_{\mathcal{T}}) \\ - \quad & clearn(lt) \\ + \quad & p_i(cap_{i,k} \mid \mathcal{T}) \cdot p(\mathcal{T}) \cdot U_i(\mathcal{T}_{type}). \end{aligned} \tag{2}$$

The first term in Equation 2 is the weighted learning gain multiplied by the likelihood of a_i encountering a_j again in the future. The second term is the cost of performing a particular type of learning. The third term is the look-ahead term corresponding to the potential utility of gaining knowledge in $cap_{i,k}$. To select a capability to learn, the agent to learn from and the learning type to employ, an agent a_i can use three strategies:

(a) Strategy 1 (Optimizer): Select the learning type, agent and capability $< lt, a_j, k >$ triplet that maximizes $U_i(lt, a_j, k)$:

$$< lt, a_j, k >^*_{opt} = \max_{lt} \left[\max_{k} \left\{ \max_{a_j} U_i(lt, a_j, cap_{i,k}) \right\} \right]$$

(b) Strategy 2 (Agent Selector): Select the agent a_j that maximizes the sum of $U_i(lt, a_j, k)$ for all capabilities k. This strategy allows an agent to learn all capabilities from the agent in $A_{\mathcal{T}}$ that is the most 'well-rounded' in its capability set.

$$< lt, a_j, - >^*_{Ag} = \max_{lt} \left[\max_{a_j} \left\{ \sum_{k} U_i(lt, a_j, cap_{i,k}) \right\} \right]$$

(c) Strategy 3 (Capability Selector): Select the capability k that maximizes the sum of $U_i(lt, a_j, k)$ for all agents a_j in $A_{\mathcal{T}}$. This strategy allows an agent to learn about the 'most qualified' capability in the team from all other agents in the team.

$$< lt, -, k >^*_{Cap} = \max_{lt} \left[\max_{k} \left\{ \sum_{a_j} U_i(lt, a_j, cap_{i,k}) \right\} \right]$$

As a result of learning, the quality of agent a_i's capability $cap_{i,k}$ gets updated by a quantity proportional to the effectiveness of the learning type, as given in the following equation:

$$qual(cap_{i,k}) \leftarrow qual(cap_{i,k}) + \Delta_{qual}(elearn(lt), Gain(lt, a_i, a_j, k)) \tag{3}$$

where $\Delta_{qual} : L_{type} \times \mathbb{R} \rightarrow \mathbb{R}^+$ is a quality update function based on the learning type. It is calculated as

$$\Delta_{qual} = elearn(lt) Gain(lt, a_i, a_j, k) \tag{4}$$

Δ_{qual} determines by how much the quality of capability $cap_{i,k}$ is changed based on the learning type, e.g., whether the increment in the quality of the capability will be more when learned by being taught than when learned through observation. In case no learning occurs for capability $cap_{i,k}$ and the agent does not use capability $cap_{i,k}$ either, the quality of the unused capability $cap_{i,k}$ decreases with time, as $qual(cap_{i,k}) \leftarrow qual(cap_{i,k}) - \gamma$, where γ is a small real value.

The overall expected utility of the agent a_i from executing task $\mathcal{T}$ of type $\mathcal{T}_{type}$ followed by learning capabilities is given by a weighted sum of its expected utility from executing the task (from Equation 1) and expected utility from learning and teaching related to the subtasks of that task (from Equation 2), as given below:

$$\begin{aligned} U_i(\mathcal{T}_{type}) \leftarrow & \; U_i(\mathcal{T}_{type}) \\ + \; & w \cdot U_i(\mathcal{T}) + (1 - w) \cdot \sum_{k \in cap_i} \sum_{a_j} U_i(lt, a_j, k), \end{aligned} \tag{5}$$

where $w \in [0, 1]$ is the weight that represents the importance of performing the task.

The algorithm used by an agent a_i in our ad-hoc collaboration environment is shown in Algorithm 2.

IV. Experimental Results

We implemented our multi-agent ad-hoc collaboration model in Repast Simphony (repast.sourceforge.net), an agent- based simulation framework. The main objective of our simulations is to demonstrate the usefulness and correctness of our learning framework and to compare it with a fixed behavior. We do this by analyzing qualities of the capabilities, number of finished subtasks and average utilities of the agents over time. In particular, in our experiments we evaluate the following hypotheses: (1) task openness parameter controls how many tasks are completed, (2) agent openness parameter impacts the quality of the task but not so much the task completion, (3) agents using a utility maximizing strategy to make a decision about learning obtain higher utility and complete more tasks than agents not using such strategies, (4) agents that try to learn all capabilities (not necessarily optimally) outperform other agents that optimize or select one capability to learn, and (5) agents that dynamically learn capabilities from other agents get higher utility and complete more tasks than agents that use a fixed learning strategy. We simulate a search-and-rescue operation scenario comprising a set of tasks

```
selectAndExecuteTasks()
Blackboard b;
foreach timestep do
    if Subtask in last timestep was completed then
        selectSubTasks(b);
        if subtask queue not empty then
            execute first subtask from subtask queue;
            if task corr. to subtask is complete then
                1: calculate utility of task using Eqn. 1;
                2: calculate learning utility and choose
                    < lt, a_j, cap_{i,k} > using Eqn. 2;
                3: update quality of learned capability using Eqn. 3;
                4: decrement quality of capabilities not used for this subtask;
                5: update util. of task type corr. to task using Eq. 5;
            end
        end
        else
            // subtask queue empty because no cap-s at desired quality to do available subtasks
            // learn the 'best' capability to perform expected future tasks
            execute steps 2-5 above;
        end
    end
    else
        continue unfinished subtask from last time step;
    end
end
```

Algorithm 2: Algorithm used by agent a_i to select and execute subtasks

that need to be accomplished during the operation. In all our simulations one task is introduced at each time step and we use 50 agents representing human members of a search-and-rescue team. The task types and the order in which tasks arrive are predetermined for our simulation purposes. We consider a maximum of 5 unique subtasks (removing debris, extracting a victim, providing medical assistance, navigation, tracking); thus there can be $251 = 31$ unique task types. Each subtask requires one type of agent capability and thus there are 5 agent capabilities. We ran our simulations for different value combinations of task openness and agent openness parameters - **task openness** $\in \{0, 0.5, 1\}$ denotes the fraction of new tasks that are introduced at the end of each time step and **agent openness** $\in \{0, 0.5, 0.9\}$ denotes the fraction of new agents entering at the end of every time step. The local satisfaction obtained by an agent for completing a subtask, $localSat(\cdot)$, is drawn from a Gaussian distribution with the mean at the quality level $qual(cap_{i,k})$ used for completing the subtask. Corresponding, the global satisfaction value, $globalSat(\cdot)$, that an agent receives after a task, in which it had participated by performing a subtask, is completed is drawn from a Gaussian distribution with the mean at the sum of the values $V_i(\tau, localSat(\cdot))$ for all subtasks $\tau \in \mathcal{T}$ associated with task $\mathcal{T}$. We assume that there is a domain-expert, external entity (human or software agent) that calculates the value of $globalSat(\cdot)$. The values of the different parameters we have used in our experiments are given in Table II. All results were averaged over 10 simulation runs.

Name	Value
η(self learning gain)	0.01
c(learning gain cap)	2
ϵ(learning gain zero-offset)	0.001
γ(decrease in quality)	0.01
$elearn(\{1,2,3,4\})$	$\{0.2, 0.4, 0.6, 0.8\}$
$clearn(\{1,2,3,4\})$	$\{0.2, 0.4, 0.6, 0.8\}$
w(wt. in Eqn. 5)	0.5
$qualThresh_{\{1,2,3,4,5\}}$	{0.7,0.5,0.4,0.3,0.2}
$c_{\tau_{i,j}}$(cost of subtask j to a_i)	$\mathcal{N}(qual(cap_{i,j}), 1)$
$globalSat(\tau_{i,j}, qual(cap_{i,j}))$	$\mathcal{N}(localSat(\tau_{i,j}, qual(cap_{i,j})), 1)$

TABLE II
PARAMETERS AND THEIR VALUES USED IN OUR SIMULATION EXPERIMENTS.

A. Effects of Task and Agent Openness Parameters

In our first set of experiments, we analyze the effect of task and agent openness on the performance of quality of tasks and completion rate of tasks for different combinations of the task and agent openness parameter values. All agents use Strategy 1 (Optimizer) to make a decision about learning capabilities. At the beginning of the simulation each agent randomly selects how many and what capabilities it possesses along with an initial quality value in $U[0, 1]$ for each selected capability.

Figure 2(a) shows changes in average quality of all agents over time for the capability that was selected by the largest number of agents. The maximum observed standard deviation for these experiments was obtained as 0.11. Figure 2(b) shows changes in average quality of all agents over time for the capability that was selected by the smallest number of agents. The maximum observed standard deviation for these experiments was obtained

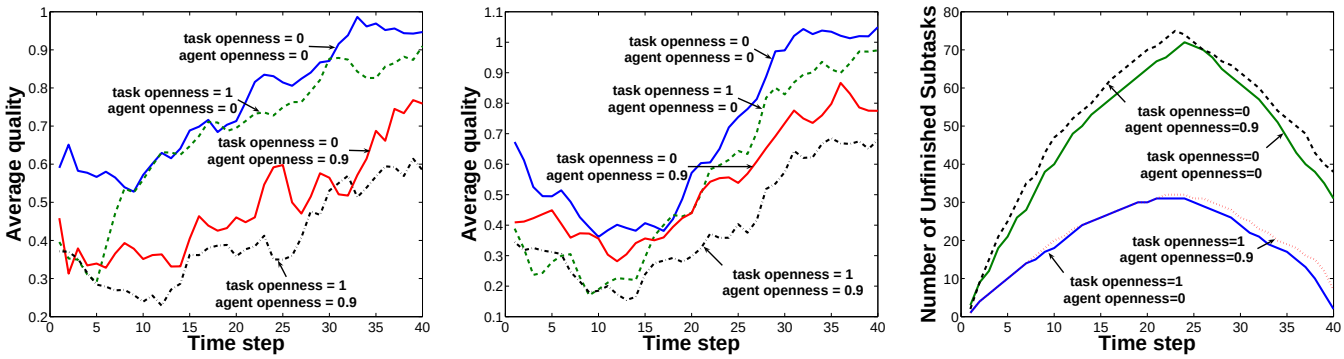

Fig. 2. (a) Average quality of the capability of tasks completed with the largest number of agents, (b) average quality of the capability of tasks completed with the smallest number of agents, (c) number of unfinished subtasks for different combinations of task and agent openness parameters.

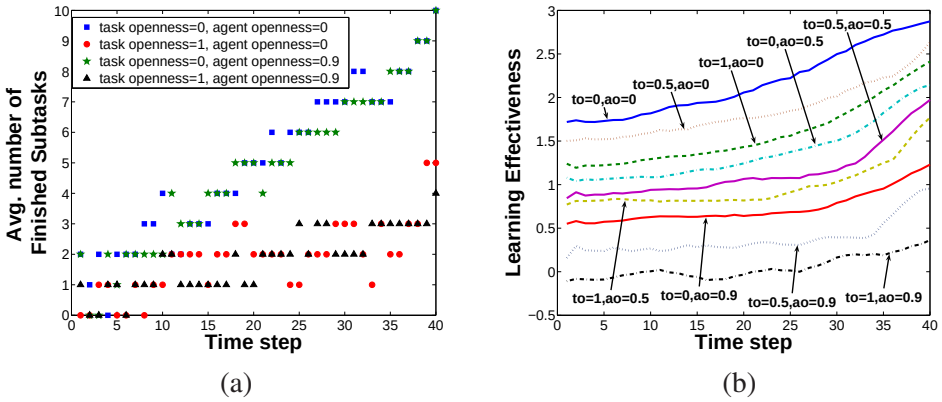

Fig. 3. (a) Average number of subtasks that are finished at each time steps, (b) Learning effectiveness for 9 different combinations of task and agent openness.

as 0.08. We observe that the quality decreases slightly initially but then it tends to increase over time as agents learn capabilities from each other. The reason for the initial decrease in quality is due to a bootstrapping period during which agents discover what capabilities are most in demand and then learn those capabilities. The bootstrapping period is also evidenced in Figure 2(c), where the number of unfinished subtasks at the end of each time step increases initially and then starts decreasing when agents acquire capabilities to perform the tasks. Also, in Figure 2(a), we observe that when agent openness and task-openness are both equal to 0, the quality of tasks performed is the highest as the environment is least dynamic. When agent-openness $= 0$, agents perform tasks with $15 - 32\%$ higher average qualities than when agent openness $= 0.9$, irrespective of the task openness value. This is because with a static agent population, the capabilities learned by the agents remain within the population and can be reused. However, varying the task openness between 0 and 1 while keeping agent openness fixed, only marginally reduces the task performance quality. Overall, these results illustrate that the agent openness parameter is more critical than task openness parameter in determining the performance quality of tasks for an ad-hoc collaboration scenario. The task openness parameter on the other hand is more crucial in determining how many tasks get completed. In Figure 2(c), we see that the number of unfinished subtasks is higher when task openness $= 0$ because agents require more time to learn capabilities so that their possessed capabilities are aligned with the tasks' capabilities.

We further analyze the above hypothesis about the effect of agent and task openness parameters on the completion rate of tasks in the following experiment where we record the number of finished subtasks at every time step averaged over the number of agents. Results are shown in Figure 3(a). We observe that when all of the tasks are of the same task type, i.e., when task openness is 0, agents complete 49% more subtasks than when all of the tasks are of different task type. We

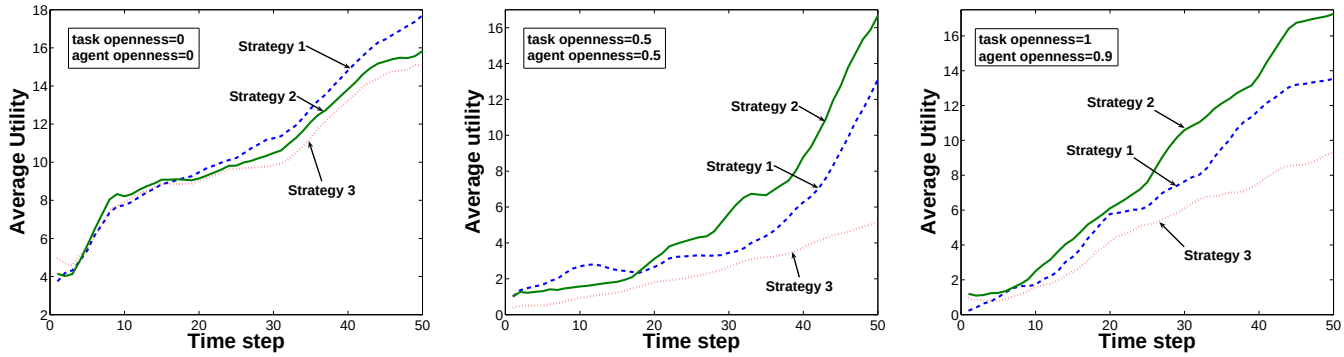

Fig. 4. Average utility of the agents for different strategies described in Section III-D that are used by the agents to make a decision about learning when (a) task and agent openness are 0, (b) task and agent openness are 0.5, (c) task openness is 1 and agent openness is 0.9.

also note that when the same agents remain throughout the simulation (agent openness $= 0$), 4% more subtasks are finished than when only 10% of the agents are the same throughout the simulation (agent openness $= 0.9$). We posit that the difference in the number of finished subtasks is not very significant when agent openness changes from 0 to 0.9 because new agents still have some capabilities and are able to perform and complete subtasks. On the other hand, task openness parameter affects the rate of completion of subtasks - a lot more subtasks are finished when task openness is 0 than when it is 1. This is because with task openness $= 0$ agents learn the capabilities for the subtasks of that one task type throughout the duration of the simulation and more agents are therefore able to perform the subtasks of that particular task type with time.

B. Analyzing costs and benefits of learning capabilities

Learning Effectiveness. Our results show that intelligently learning capabilities based on task requirements produces higher agent utilities for most scenarios. This is especially true in more static environments (with low task and/or agent openness), where $LE > 1$. However, as shown in Figure 3(b), for more dynamic environments, the benefit of learning capabilities may diminish. When the task and agent openness values are very high, LE is initially negative and does not reach 1 up to the end of the experiments. This may happen because in highly dynamic environments, the required capabilities change rapidly as the tasks that are introduced at each time step are almost all new tasks. Meanwhile, the agents that are there from which to learn capabilities are mostly 'new', non-expert agents as only 10% of the agents persist from one time step to the next. Therefore, the quality values of an agents capabilities may take considerable time to reach a level at which their utility from task execution is high enough to offset the expenditure for learning. As long as this offset does not happen, the utility from learning is negative. Consequently, we can conclude that the openness parameters for tasks and agents play a crucial role in determining the performance and convergence in ad-hoc coalition formation, and thus important factors to consider when studying ad-hoc coalitions.

Learning Population Makeup. For our next experiment we consider a setting with different types of agents. We allow for two types of agents - zero intelligent and intelligent agents, where intelligent agents use strategy 1 to make a decision about learning while zero intelligent agents do not use utility maximizing strategies to make decision about learning, instead they randomly select the capability they want to learn and the agent they want to learn it from. We vary agent population into setups where (1) all agents are zero intelligent, (2) all agent are intelligent, and where (3) the agent population is divided evenly into zero-intelligent and intelligent agents. We keep the other parameters fixed with: task openness $=$ agent openness $= 0.5$, no. of agents $= 50$, and all agents are intelligent. The utility averaged over all agents in the population is shown in Figure 5(a) and (b). In Figure 5(a), we observe that the zero intelligent agents get 38% less utility on average and complete 60% less tasks on average than the intelligent agents. In Figure 5(b), we observe that when the population comprises of 50% intelligent and 50% zero intelligent agents, the zero-intelligent agents get more utility than in the all-zero intelligent case. This indicates that being in a population with agents that intelligently learn capabilities improve utility for zero intelligent agents too as more tasks get completed by the intelligent agents and the zero intelligent agents derive the utility from those completed tasks. The number of subtasks that are finished at each

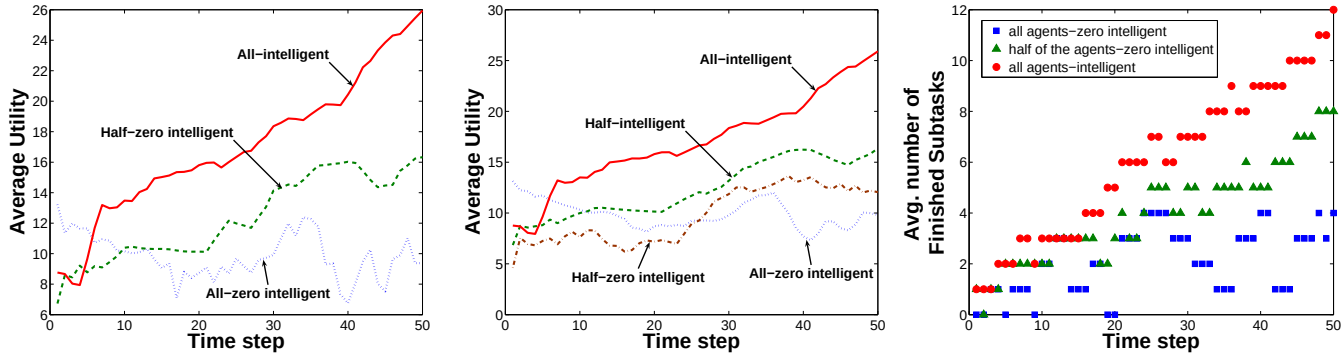

Fig. 5. (a) Avg. utility of agents in a population when the population has all intelligent(All-intelligent), all zero-intelligent agents(All-zero intelligent) or 50% intelligent and 50% zero-intelligent(Half-intelligent) (b) Avg. utility of agents for the same settings as in (a) but showing the division of utilities between intelligent and zero intelligent agents in the 50% intelligent(Half-intelligent) and 50% zero-intelligent(Half-zero intelligent) population, (c) Avg. no. of finished subtasks for the different agent population setups.

time step averaged over all agents in shown in Figure 5(c). Overall, we observe that intelligent agents are able to get more utility than zero intelligent agents by using strategic decisions about learning capabilities.

Openness-based Learning Strategies. Next, we analyze the performance of the three different capability learning strategies given in Section III-D. Each setting has 50 agents and all agents use the same strategy in one simulation. The task openness and agent openness parameters are varied over $0, 0.5$, and $1(0.9)$ respectively. All other parameters are retained from the previous experiments. Figures 4 (a)-(c) show the utility averaged over all agents for the different strategies. We observe that Strategy 2 (Capability Selector), where an agent first selects the most suitable or expert agent to learn from and then learns all the other capabilities (opportunistically) from it, gives the highest utility when task and agent openness are 0.5 and 1. Strategy 3 (Agent Selector), where the agent learns the same capability from all agents gives the lowest utility. These results indicate that it is a better strategy to learn all capabilities "in a bundle" from one expert even though not all capabilities learned will be optimal rather than to try to become an expert in each capability by learning it separately from each corresponding expert respectively. This indication is more pronounced when the environment is more dynamic (task and agent openness are 1 and 0.9 resp.): the difference between the agent utilities from Strategies 2 and 3 increases by about 25% compared to a less dynamic environment (task and agent openness are both 0). Overall, these results indicate that being all-rounded in terms of capabilities is also a better game plan, and it also pays off better as the dynamic nature in the environment increases.

Dynamic vs. Fixed Learning Strategies. For our final set of experiments, we compare the benefit of dynamically learning capabilities from other agents using our framework vs. a fixed learning strategy. We consider two populations of agents - agents with a *dynamic* strategy use Strategy 1 to select the capability to learn and which other agent to learn from, while agents with a *fixed* strategy select each capability to learn with a fixed probability of $\frac{1}{|Cap|} = \frac{1}{5}$. We consider two agent populations, one where all agents are dynamic and compare it with a population where all agents use fixed strategy. We vary the task openness parameter and set agent openness $= 0$. All other parameters are kept at the values from the previous experiment. The average utility for all agents for different values of task openness is shown in Figure 6(a-c), and the number of subtasks that are finished at each time step averaged over all agents for different values of task openness is shown in Figure 6(d-f). We observe that fixed strategy agents get 64% less utility on average and complete 52% less tasks on average than the dynamic strategy agents when task openness is 0. This is because fixed agents waste resources by learning different capabilities other than just the necessary ones in the stable environment where tasks' types do not change. Fixed strategy agents do better when task openness is 1, but they still get 47% less utility on average and complete 26% less tasks on average than the dynamic agents. These results clearly illustrate that dynamic agents are able to make smart decisions about learning capabilities as well as improve their own utilities, while fixed agents make arbitrary learning decisions and adversely affect task performance in the environment as well as their own utilities.

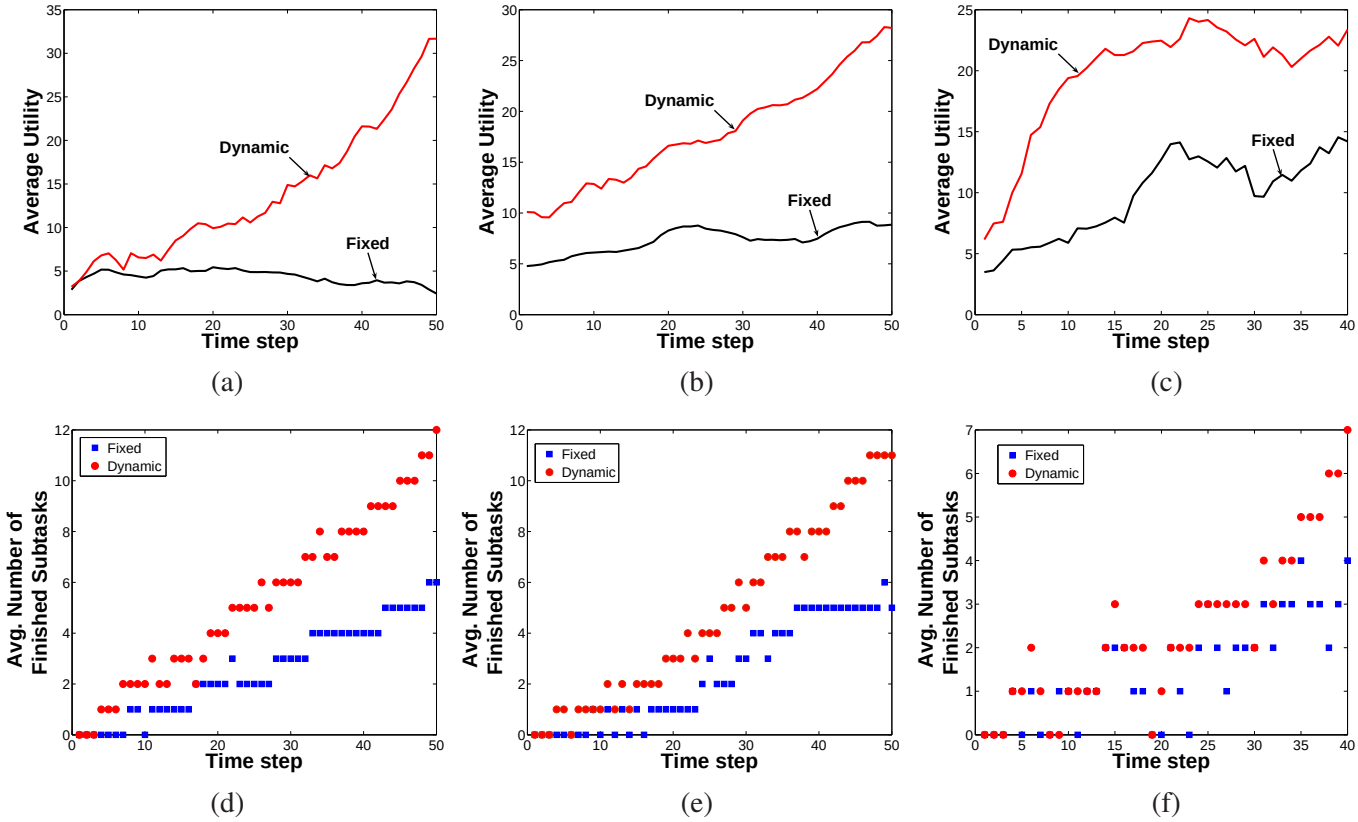

Fig. 6. Average utility of the agents when task openness is (a) 0, (b) 0.5, (c) 1, and the average number of finished subtasks when task openness is (d) 0, (e) 0.5, (f) 1 for different agent populations.

V. Conclusion and Future Work

In this paper, we described a framework for agents to perform tasks in an ad-hoc collaborative manner while strategically learning capabilities to perform the tasks. Novel features of our model account for agent and task openness within a multi-agent learning and teaching model. Our experimental results indicate that agents learn and perform the best in more stable environments, while in more dynamic environment strategic decision-making by the agents to learn capabilities improves the agent utilities and their task performance. Finally, we observed that even if only a part of the agent population strategically learns capabilities, they are able to improve the overall task performance in the environment, and, consequently, increase the utility of the less intelligent agents in the population. This result has an important repercussion - in environments where learning capabilities is a costly operation, only a fraction of the population can learn capabilities strategically, but still improve the task performance of the environment and contribute to the utility of the entire agent population. Besides search-and-rescue scenarios, we foresee our work to be applicable to military operations, where soldiers having different capabilities have to work together on one operation; to task force collaborations, where experts from different institutions use their skills and collaborate in response to a particular crisis or event; to distance learning students working on group projects; to business companies where different groups inside the company have to work together on a specific goal, etc.

Future work includes investigating other stress factors in ad-hoc collaboration environments such as priorities of tasks, noise, tight time constraints, and disappearing team members (i.e., team members who leave an ad hoc team due to other emergency commitments) and determining optimal values for the learning gain cap and zero offset parameters in Table II. We also plan to investigate different learning models and the use of different communication protocols (both direct and indirect) that impact the learning effectiveness and costs. Another relevant direction we plan to investigate is designing more complex protocols for disbursing the global satisfaction value that the agent receives after the whole task is completed using concepts from coalitional game theory such as the core and the Shapley value [17]. We also look into more varied task types where capabilities of each agent is a significant subset of the total number of capabilities. And finally, we plan to

conduct more extensive simulations to get a better idea of how selective agents can be in learning and teaching by increasing the number of agent capabilities, so that each agent can only be good at a few of them, and by having more diverse task types.

VI. Acknowledgments

This research has been sponsored as part of the COMRADES project funded by the Office of Naval Research, grant number N000140911174.

References

[1] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein, "Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination," in *Proceedings of the 24th Conference on Artificial Intelligence*, 2010, pp. 1504–1509.

[2] R. Kildare, "Ad-hoc online teams as complex systems: agents that cater for team interaction rules," in *Proceedings of the 7th Asia-Pacific Conference on Complex Systems*, 2004, pp. 282–291.

[3] N. Khandaker and L.-K. Soh, "Formation and scaffolding human coalitions in i-minds - a computer-supported collaborative learning environment," in *Proceedings of the Workshop on Agent-Based Systems for Human Learning and Entertainment*, 2007, pp. 64–75.

[4] L. Vygotsky, *Mind in Society*. Boston, MA: Harvard University Press, 1978.

[5] B. Wilsker, "Study of Multi-Agent Collaboration Theories," in *USC Tech Report no. ISI/RR-96-449*, 1996.

[6] P. Stone and S. Kraus, "To teach or not to teach?: decision making under uncertainty in ad hoc teams," in *Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010, pp. 117–124.

[7] N. Agmon and P. Stone, "Leading ad hoc agents in joint action settings with multiple teammates," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012, pp. 341–348.

[8] S. Barrett and P. Stone, "An analysis framework for ad hoc teamwork tasks," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012, pp. 357–364.

[9] A. Stefano and R. Subramanian, "Comparative evaluation of mal algorithms in a diverse set of ad hoc team problems," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012, pp. 349–356.

[10] K. Genter, N. Agmon, and P. Stone, "Role-based ad hoc teamwork," in *Proceedings of the Plan, Activity, and Intent Recognition Workshop at the Twenty-Fifth Conference on Artificial Intelligence (PAIR-11)*, 2011. [Online]. Available: http://www.cs.utexas.edu/users/ai-lab/?PAIR11-katie

[11] S. Liemhetcharat and M. Veloso, "Modeling mutual capabilities in heterogeneous teams for role assignment," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3638–3644.

[12] S. Wua, H. Ghenniwaa, Y. Zhanga, and W. Shena, "Personal assistant agents for collaborative design environments," *Computers in Industry*, vol. 57, no. 8-9, pp. 732 – 739, 2006.

[13] K. Myers, P. Berry, J. Blythe, K. Conley, M. Gervasio, D. McGuinness, D. Morley, A. Pfeffer, M. Pollack, and M. Tambe, "An intelligent personal assistant for task and time management," *AI Magazine*, vol. 28, no. 2, pp. 47–61, 2007.

[14] P. Brusilovsky, S. Sosnovsky, and O. Shcherbinina, "User modeling in a distributed e-learning architecture," vol. 3538, pp. 387–391, 2005.

[15] A. Baylari and G. Montazer, "Design a personalized e-learning system based on item response theory and artificial neural network approach," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8013 – 8021, 2009.

[16] M. Wooldridge, *An Introduction to Multiagent Systems*. UK: Wiley, 2009.

[17] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations*. Cambridge University Press, 2009.

[18] A. Inaba, T. Supnithi, M. Ikeda, R. Mizoguchi, and J. Toyoda, "How can we form effective collaborative learning groups," in *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, 2000, pp. 282–291.