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

Janyl Jumadinova[1], Prithviraj Dasgupta[1], Leen-Kiat Soh[2]
[1]Department of Computer Science, University of Nebraska at Omaha
[2]Department of Computer Science and Engineering, University of Nebraska - Lincoln
E-mail:{jjumadinova, pdasgupta}@unomaha.edu, lksoh@cse.unl.edu

***Abstract*—We consider the problem of distributed collaboration among multiple agents to perform tasks in an ad-hoc setting. Because the setting is ad-hoc, the agents could be programmed by different people and could potentially have different task selection and task execution algorithms. We consider the problem of decision making by the agents within such an ad-hoc setting so that the overall utility of the agent society can be improved. In this paper we describe an ad-hoc collaboration framework where each agent strategically selects capabilities to learn from other agents which would help it to improve its expected future utility of performing tasks. Agents use a very flexible, blackboard-based architecture to coordinate operations with each other and model the dynamic nature of tasks and agents in the environment using two 'openness' parameters. Experimental results within the Repast agent simulator show that by using the appropriate learning strategy, the overall utility of the agents improves considerably.**[1]

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.

Recently, in [1] the authors have emphasized the advantages of using multi-agent systems to automate human ad-hoc collaboration problems in an efficient manner. Using human ad-hoc team formation [2] as a basis, they have prescribed three high-level technical directions to solve the ad-hoc collaboration problem. One of the frameworks suggested in [1] is the teacher-learner framework. In this paper we extend the teacher-learner approach for ad-hoc collaboration by developing a task-capability based multi-agent framework within which agents can use different strategies to learn task capabilities from each other. The learning and task execution operations of each agent are de-coupled from each other and agents need not necessarily learn capabilities related only to the tasks they have recently performed. Instead, they can select the 'best' capability to acquire by learning and the 'best' teacher (agent) to learn it from. Our proposed learning framework is based on principles from human learning theory [3], [4]. In addition, each agent employs two parameters called *agent openness* and *task openness* to model the dynamic nature of the agents and tasks in the environment. This enables agents to calculate the expected utility of learning task capabilities from other agents more accurately so that they can make better learning-related decisions that improve their overall utility. 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 learning get more utility than the agents that do not use learning. We also find that the agents in more open environments are able to learn and complete tasks more effectively than the agents in more closed environments. We also show that agents that use our learning technique outperform the agents that learn capabilities for performing tasks with a fixed probability.

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

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

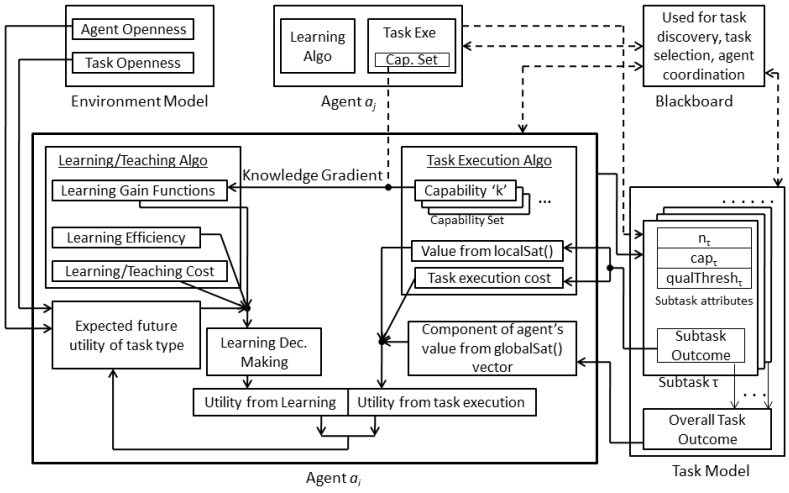


Figure 1. Task Execution and capability learning components showing agents a_i and a_j that work together on subtask τ within the ad-hoc multi-agent collaboration model.

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. In [6], the authors consider the problem of a single ad-hoc team player leading other teammates that use a best response strategy to select their actions, to the optimal (cost-minimizing) joint action. In [7], 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 [8], 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 [7], 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.

III. Multi-agent Ad-hoc Collaboration Model

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, as illustrated in Figure I. The functionality of each of these models is explained below:

A. Task Execution 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 sub-tasks. 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.

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 [9]. 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 τ is associated with three parameters, the capability cap_τ required for it, the minimum number of agents, n_τ with that capability required to perform it and the minimum quality threshold $qualThresh_\tau$ of the capability of each agent performing it. Each agent a_i uses a distributed, self-selection process given in Algorithm 1 to select sub-tasks. When an agent a_i examining the task has a capability higher than those of the currently selected agents for the task, the agent a_j with the lowest capability is de-selected for the task and informed about this de-selection. At the end of this selection process, the agents with maximal quality of the capability required of each subtask are allocated to perform the task. Agents that are allocated to perform the same subtask coordinate with each other using some coordination protocol [9] to perform their actions on the task. [2]

```
selectSubTasks(Blackboard b)
Set τ_i; // subtask list for agent a_i
Set τ.selected; // set of agents ids selected to perform τ
Set τ.quals; // set of qual(cap()) values of agents performing τ
double τ.minQual; // min. value of qual(cap()) among agents
performing τ
foreach subtask τ of T in b do
    for k ← 1 to | cap_i | do
        // check if a_i is qualified to perform subtask τ
        if (cap_τ = cap_{i,k}) and (qual(cap_{i,k}) > qualThresh_τ)
        then
            // check if a_i is better qualified than any agent already
            selected
            if (| τ.selected |= n_τ) and
            (qual(cap_{i,k}) > τ.minQual) then
                j ← agent id with quality τ.minQual
                τ.selected ← τ.selected − {j}
                τ.quals ← τ.quals − {qual(cap_{j,k})}
                send message to agent j to remove τ from its
                subtask list τ_j
            end
            τ.selected ← τ.selected ∪ {a_i}
            τ_i ← τ_i ∪ τ
            τ.quals ← τ.quals ∪ {qual(cap_{i,k})}
            τ.minQual ← min(τ.quals)
        end
    end
    if received message to remove a subtask τ then
        τ_i ← τ_i − τ
    end
end
```

Algorithm 1: Algorithm used by agent a_i to select subtasks

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 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, $globalSat : \times_{\tau \in \mathcal{T}} \quad \tau \times [0,1]^{n_\tau} \rightarrow [0,1]$, is used to denote how satisfactorily the task was completed. It is calculated by an external "reviewer" entity that assesses the outcome of the overall task based on the outcome of the subtasks. 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., $V_i(\mathcal{T}) = \sum_{\tau \in \mathcal{T}} V_i(\tau, localSat(\cdot)) + V_i(\mathcal{T}, globalSat(\cdot))$. 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, $C_i(\mathcal{T}) = \sum_{\tau_{i,j} \in \mathcal{T}} c_{\tau_{i,j}}$. Each agent a_i has a utility for task $\mathcal{T}$ given by the following utility function:

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

B. 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 [10], 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.

[2]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 [9] could be used for performing these activities depending on the application domain.

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[3] for capability k to be proportional to the capability difference between them given by $qual(cap_{i,k}) - qual(cap_{j,k})$. 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.

C. Capability Learning Strategy

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

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

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 uses a strategy called *Optimizer*:

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

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

As a result of learning, the quality of agent a_i's capability $cap_{i,k}$ gets updated by a quantity proportional to the ef-

[3]In the rest of the paper we assume $a_j \in A_{\mathcal{T}} \setminus \{a_i\}$

fectiveness of the learning type, as given in the following equation:

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

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

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:

$$U_i(\mathcal{T}_{type}) \leftarrow U_i(\mathcal{T}_{type}) + w \cdot U_i(\mathcal{T}) + (1-w) \cdot \sum_{k \in cap_i} \sum_{a_j} U_i(lt, a_j, k) \quad (4)$$

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

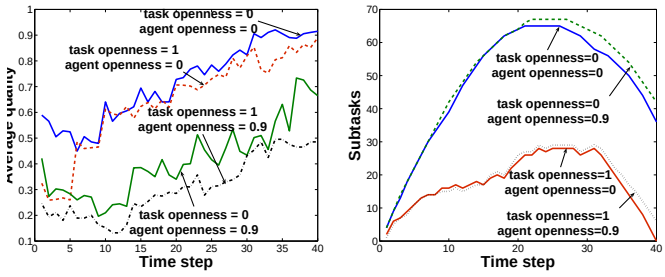
```
selectAndExecuteTasks()
Blackboard b;
foreach timestep do
    if Subtask in last timestep was completed then
        selectSubTasks(b);
        if subtask queue not empty then
            execute first subtask from subtask queue;
            if task corr. to subtask is complete then
                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.
                4;
            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 sub-tasks

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 all of our simulation setups one task is introduced at each time step and we use 50 agents. The task types and the order in which tasks arrive are predetermined. We consider a maximum of 5 unique subtasks, thus there can be 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. All results were averaged over 10 simulation runs. The values of different parameters we have used in our simulation is given below. In our first set of experiments,

Name	Value
η(self learning gain)	0.01
c(learning gain cap)	2
ϵ (learning gain zero-offset)	0.001
$elearn(\{1,2,3,4\})$	$\{0.2, 0.4, 0.6, 0.8\}$
$clearn(\{1,2,3,4\})$	$\{0.2, 0.4, 0.6, 0.8\}$
w (wt. in Eqn. 4)	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)	$\mathcal{N}(qual(cap_{i,j}), 1)$
$globalSat(\tau_{i,j}, qual(cap_{i,j}))$	$\mathcal{N}(localSat(\tau_{i,j}, qual(cap_{i,j})), 1)$

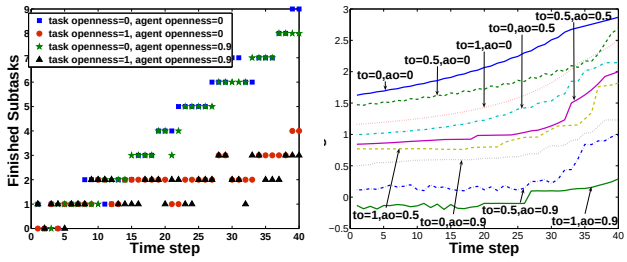


(a) (b)

Figure 2. (a) Average quality of the capability of tasks completed, (b) number of unfinished subtasks for different combinations of task and agent openness parameters.

we analyze the effect of task and agent openness on the performance quality of tasks and completion rate of tasks for different combinations of the task and agent openness parameter values. All agents use Optimizer Strategy 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 0.12. 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(b), 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. 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(b), 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 39% more subtasks than when all of the tasks are of different task type. We also note that when the same agents remain throughout the simulation (agent openness $= 0$), 3% more subtasks are finished than when only 10% of the agents are the same throughout the simulation (agent openness $= 0.9$). In our next set of



(a) (b)

Figure 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.

experiments, we analyze the effect of learning capabilities on task performance. We use a parameter called *learning effectiveness* (LE) that gives the ratio between utilities of agents when they learn capabilities using Strategy 1 and when they select tasks to perform at random without learning any capabilities. Figure 3(b) shows the learning effectiveness for different combinations of task and agent openness values. Our results validate that learning capabilities based on task requirements produces higher agent utilities ($LE > 1$) for most scenarios. However, for more dynamic environments, the benefit of learning capabilities diminishes. 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 happens because in highly dynamic environments, the required capabilities change rapidly as all new tasks are introduced at each time step.

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. 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). 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.

References

[1] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein, "Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination," in *Proc. 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 *Proc. 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] N. Agmon and P. Stone, "Leading multiple ad hoc teammates in joint action settings," in *Proceedings of the Interactive Decision Theory and Game Theory Workshop*.

[7] K. Genter, N. Agmon, and P. Stone, "Role-based ad hoc teamwork," in *Proc. Plan, Activity, and Intent Recognition Workshop*.

[8] S. Liemhetcharat and M. Veloso, "Modeling mutual capabilities in heterogeneous teams for role assignment," in *Proceedings of IROS*, 2011, pp. 3638–3644.

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

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