Allocating Training Instances to Learning Agents that Improve Coordination for Team Formation

Somchaya Liemhetcharat[1] and Manuela Veloso[2]

[1]Institute for Infocomm Research, A*STAR, Singapore
`liemhet-s@i2r.a-star.edu.sg`
[2]Computer Science Department, Carnegie Mellon University, Pittsburgh, USA
`veloso@cs.cmu.edu`

Abstract. Agents can learn to improve their coordination with their teammates and increase team performance. We are interested in forming a team, i.e., selecting a subset of agents, that includes such learning agents. Before the team is formed, there are finite training instances that provide opportunities for the learning agents to improve. Agents learn at different rates, and hence, the allocation of training instances affects the performance of the team formed. We focus on allocating training instances to learning agent pairs, i.e., pairs that improve coordination with each other, with the goal of team formation. We formally define the learning agents team formation problem, and compare it with the multi-armed bandit problem. We consider learning agent pairs that improve linearly and geometrically, i.e., the marginal improvement decreases by a constant factor. We contribute algorithms that allocate the training instances, and compare against algorithms from the multi-armed bandit problem. In extensive simulations, we demonstrate that our algorithms perform similarly to the bandit algorithms in the linear case, and outperform them in the geometric case, thus illustrating the efficacy of our algorithms.

1 Introduction

Multi-agent teams have been applied to various domains, such as task allocation and multi-agent planning. Typically, the capabilities of the agents are fixed and assumed to be known *a priori*, and the performance of a team is the sum of the single-agent capabilities. The Synergy Graph model was recently introduced, where team performance depends on single-agent capabilities and the pairwise compatibility among the agents [4]. The single-agent capabilities and pairwise compatibility were assumed to be fixed and initially unknown, and learned from observations, with the goal of forming a team after learning the team performance model. In this work, we use the term *agents* to refer to physical robots, simulated robots, and software agents.

What if some agents are learning agents, i.e., they are capable of learning to improve their coordination with their teammates? We are interested in team formation, i.e., selecting a subset of agents, with such learning agents. Learning agents, which we detail in the related work section, improve coordination by modeling their teammates and varying their behaviors to maximize the team performance. We consider *learning agent pairs* that improve their coordination with each other. By doing so, we encapsulate pairs of learning agents that simultaneously learn, as well as pairs consisting of a learning agent and a regular agent.

In this paper, we formally define the learning agents team formation problem, where the goal is to form a multi-agent team, and there is a fixed number of training instances available before the team is formed. Each training instance is allocated to a learning agent pair to improve their coordination. A motivating example is from sports, where a coach has limited opportunities to train his team before the actual game. The coach allocates training instances to pairs (e.g., a pair that practices passing the ball upfield in soccer), and the pair improve their coordination. When all the training is done, the coach selects which members form the team.

Hence, the allocation of training instances has a large impact on the performance of the formed team. In particular, a team with low performance before training may outperform all other teams after training, if it is comprised of learning agent pairs that have high learning rates, i.e., large improvements in coordination per training instance. However, the heterogeneous learning rates of learning agent pairs are initially unknown, and have to be learned from observations after each training instance. Thus, solving the learning agents team formation problem requires balancing exploring and exploiting, while modeling the learning rates and keeping the end goal of team formation in mind.

We consider scenarios where learning agent pairs improve linearly, and where learning agent pairs improve geometrically, i.e., the marginal improvement decreases by a constant factor $0 < \gamma < 1$ after each training instance. We consider the optimal allocation of training instances if the learning rates are known, and introduce algorithms that allocate training instances while modeling the learning rates, in both the linear and geometric scenarios. There are parallels between the learning agents team formation problem and the multi-armed bandit problem, namely by considering each learning agent pair as an arm and a training instance as pulling an arm. However, a key difference is that our goal is to form the team with optimal performance, while the goal of the multi-armed bandit problem is to maximize the cumulative sum of rewards. We compare our algorithms with the upper confidence bound and Thompson sampling algorithms from the multi-armed bandit problem.

We conduct experiments in simulation, varying the number of learning agent pairs and training instances, and whether learning agent pairs improve linearly or geometrically. We demonstrate that the algorithms we contribute perform similarly to the bandit algorithms when improvements in coordination are linear, and outperform them in the geometric scenario. Our algorithms perform close to optimal without having *a priori* information about the heterogeneous learning rates, thus illustrating the efficacy of our algorithms in the learning agents team formation problem.

The layout of our paper is as follows: Section 2 discusses related work and how our work builds upon prior research on learning agents. Section 3 formally defines the learning agents team formation problem and gives an overview of our approach. Sections 4 and 5 consider learning agent pairs that improve linearly and geometrically. Section 6 presents our experiments and results, and Section 7 concludes.

2 Related Work

Learning agents have been studied in various fields, such as game theory, multi-agent planning, and reinforcement learning, e.g., [6, 10, 8]. The ad hoc problem was recently introduced, where an autonomous agent learns to collaborate with previous unknown teammates [7]. An ad hoc agent can lead multiple teammates to select the optimal joint action [1], and an ad hoc agent can also model its teammates and change its policy to optimize a team goal [3]. While we discuss ad hoc agents in detail, our work is not specific to ad hoc agents and is applicable to general learning agents that learn to improve their coordination with teammates.

We are interested in modeling learning agents for team formation. Our perspective is different from other learning agents research in that they typically focus on how an agent can learn and adjust its behaviors based on its teammates, while our focus is on modeling the impact of learning agents on the team, and allocating training instances for the learning agents to improve their coordination with teammates, and hence improve team performance. We use the recently-introduced Synergy Graph model [4, 5] to compute team performance, where team performance is beyond the sum of single-agent capabilities, but also depends on pairwise compatibilities.

There are similarities between the learning agents team formation problem and the multi-armed bandit problem, which we detail in the next section. We consider two algorithms from the multi-armed bandit literature: the upper confidence bound (UCB) [2] and Thompson sampling (TS) [9]. Each arm in the multi-armed bandit problem has an unknown probability p that is estimated, and UCB pulls the arm with the highest upper confidence bound on p. TS draws a sample for each arm based on its estimated distribution, and pulls the arm with the highest sample. We compare our algorithms for the learning agents team formation problem against the UCB and TS algorithms.

3 Problem Definition and Our Approach

In this section, we formally define the learning agents team formation problem, give an overview of our approach, and compare the learning agents team formation problem with the multi-armed bandit problem.

3.1 Learning Agents Team Formation Problem

We are interested in team formation, i.e., selecting a subset of agents, in order to maximize a performance function. We begin with the set of agents and the definition of a team:

Definition 1. *The* ***set of agents*** *is $\mathcal{A} = \{a_1, \ldots, a_N\}$, where each $a_i \in \mathcal{A}$ is an agent.*

Definition 2. *A* ***team*** *is any subset $A \subseteq \mathcal{A}$.*

The performance of a team depends on its composition, and in this work, we modify the Synergy function of the Synergy Graph model [4]:

Definition 3. *The* ***performance*** *$P(A)$ of a team A is:*

$$P(A) = \frac{1}{\binom{|A|}{2}} \sum_{\{a_i, a_j\} \in A} P_2(a_i, a_j), \text{ such that}$$
$$P_2(a_i, a_j) = \phi_{i,j} \cdot (C_i \; + \; C_j)$$

where $\phi_{i,j} \in \mathbb{R}^+$ is the coordination level between a_i and a_j, and C_i, C_j are Normally-distributed variables representing a_i and a_j's capabilities at the task.

When agents learn to perform better over time, there are two possible reasons: the agent learns about the *task* and improves its capability C_i at the task; or the agent learns to *coordinate* with its teammates better and improves $\phi_{i,j}$. We are interested in the latter, where agents learn to coordinate better with their teammates. We use the modified Synergy function to compute team performance for two main reasons. First, the performance of a team goes beyond the sum of single-agent capabilities, i.e., the capabilities of an agent pair are weighted by the coordination between them. Second, improvements in coordination are modeled with changes in $\phi_{i,j}$.

We focus on *pairs* of agents that learn to improve their coordination. By doing so, we focus on the improvement of the pair, without doing credit assignment on which of the pair (or both) is actually learning. Research on ad hoc agents, e.g., [3], demonstrated that changing the behavior of an agent in response to a teammate improves team performance. An ad hoc agent a_i would be represented by considering all agent pairs that include it (e.g., $\{a_i, a_j\}, \{a_i, a_k\}$). Our formulation thus encompasses ad hoc agents, and other learning agents, such as pairs of agents that can only improve performance with each other and no other agents. We use the term *learning agents* to refer to agents that learn to coordinate better with teammates.

We now define a learning agent pair:

Definition 4. *A* ***learning agent pair*** *is a pair of agents $\{a_i, a_j\} \in \mathcal{A}^2$. The* ***set of learning agent pairs*** *is $\mathcal{L} \subseteq \mathcal{A}^2$.*

Learning agent pairs improve their performance when they are allocated training instances, which we define next:

Definition 5. *A **training instance** $k \in \{1, \ldots, K\}$ is an opportunity for a learning agent pair $\{a_i, a_j\} \in \mathcal{L}$ to improve its coordination.*

The coordination of a learning agent pair increases after they are allocated training instances:

Definition 6. *The **coordination** of a learning agent pair $\{a_i, a_j\} \in \mathcal{L}$ after the k^{th} training instance is $\phi_{i,j}^{(k)}$.*

Note that if the training instance k was allocated to $\{a_i, a_j\} \in \mathcal{L}$, then $\phi_{i,j}^{(k)} > \phi_{i,j}^{(k-1)}$, otherwise $\phi_{i,j}^{(k)} = \phi_{i,j}^{(k-1)}$.

Training instances are allocated to learning agent pairs, and observations are obtained:

Definition 7. *An **observation** $o_{i,j} \sim P_2(a_i, a_j)$ is obtained for each training instance that is allocated to the learning agent pair $\{a_i, a_j\} \in \mathcal{L}$.*

Since $\{a_i, a_j\}$ are learning, $\phi_{i,j}$ increases as a function of the number of training instances allocated to $\{a_i, a_j\}$, and $o_{i,j}$ similarly increases on expectation.

There are $K > 0$ training instances, and the goal is to form the optimal team of given size n^* after K instances:

Definition 8. *The **optimal team** A_K^* is the team of size n^* with the highest mean performance after K training instances.*

Since learning agent pairs improve their coordination given training instances, the performance of a team $A \subseteq \mathcal{A}$ at the end of the training instances depends on the number of learning agent pairs in A, and the number of training instances each learning agent pair is allocated out of K.

3.2 Overview of Approach

We use the modified Synergy function to model the performance of multi-agent teams [4]. However, our approach is general and applicable to other multi-agent team models:

1. We model the coordination of learning agent pairs as $\phi_{i,j}^{(k)} = \phi_{i,j}^{(0)} + F_{i,j}(k_{i,j}, l_{i,j})$, where:
   - $\phi_{i,j}^{(0)}$ is the initial coordination level of $\{a_i, a_j\}$;
   - $F_{i,j} : \mathbb{Z}_0^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ is the coordination gain function (we consider $F_{i,j}$ being a linear or geometric function);
   - $k_{i,j} \leq k$ is the number of training instances allocated to $\{a_i, a_j\}$ after $k \leq K$ training instances;
   - $l_{i,j}$ is an initially-unknown learning rate of $\{a_i, a_j\}$;
2. We iteratively allocate training instances using estimates of $l_{i,j}$;
3. We use the observations $o_{i,j}$ after each training instance to improve our estimate of $l_{i,j}$.

We assume that the capability C_i of all agents $a_i \in \mathcal{A}$, coordination gain function $F_{i,j}$ and initial coordination $\phi_{i,j}^{(0)}$ of all $\{a_i, a_j\} \in \mathcal{L}$, and coordination $\phi_{\alpha,\beta}$ of non-learning agent pairs $\{a_\alpha, a_\beta\} \notin \mathcal{L}$ are known *a priori*. The only unknowns are $l_{i,j}$.

3.3 Comparison with Multi-Armed Bandits

There are similarities between the learning agents team formation problem and the multi-armed bandit problem, where learning agent pairs are arms in the bandit problem:

1. There are a fixed number of trials (training instances);
2. Each trial improves the estimate of $l_{i,j}$;
3. There is an optimal allocation of the K trials if all $l_{i,j}$ were known.

However, there is a key difference between the two problems: the goal of the multi-armed bandit problem is to maximize the **cumulative sum of rewards**, while the goal of the learning agents team formation problem is to maximize the **mean performance of a team after the K trials**.

Pulling an arm in the multi-armed bandit problem always improves the final score on expectation. In the learning agents team formation problem, assigning an agent pair a training instance improves their coordination, but may not affect the final team's score. For example, if the agent pair $\{a_i, a_j\}$ received $k_{i,j} \leq K$ training instances, but the team A that is formed does not contain the pair $\{a_i, a_j\}$, then the $k_{i,j}$ training instances did not add to the final score.

4 Learning Agents that Improve Linearly

In this section, we consider learning agent pairs that improve their coordination linearly: $\phi^{(k)}_{i,j} = \phi^{(0)}_{i,j} + F_{i,j}(k_{i,j}, l_{i,j})$, where $F_{i,j}(k_{i,j}, l_{i,j}) = k_{i,j} \cdot l_{i,j}$. First, we consider the optimal allocation of K training instances assuming all $l_{i,j}$ are known. Next, we adapt two algorithms from the bandit literature to the learning agents team formation problem, and contribute an algorithm that approximates the optimal allocation. We consider non-linear (i.e., geometric) improvement in coordination in the next section, and introduce other algorithms that are applicable to the geometric case. We use a Kalman filter to provide estimates $\hat{\phi}^{(k)}_{i,j}$ and $\hat{l}_{i,j}$ of $\phi^{(k)}_{i,j}$ and $l_{i,j}$ respectively, and the Kalman filter is updated with new observations $o_{i,j}$.

4.1 Computing the Optimal Allocation

Suppose that $\forall \{a_i, a_j\} \in \mathcal{L}$, $l_{i,j}$ is known. To compute the optimal allocation, we iterate through every possible team A, and compute the allocation of the K training instances given A. The allocation k_{A^*} corresponding to the team A^* with the maximum score is then the optimal allocation.

When there are no learning agent pairs in A, the allocation does not matter. Otherwise, the optimal allocation is to pick the best learning agent pair in A and allocate all K training instances to it, as shown below:

Theorem 1. *$k_A = (\{a_i, a_j\}, \ldots, \{a_i, a_j\})$ is the optimal allocation of training instances, where $\{a_i, a_j\} = argmax_{\{a_\alpha, a_\beta\} \in \mathcal{L} \cap A^2}(l_{\alpha,\beta}(\mu_\alpha + \mu_\beta))$, where $C_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and $C_j \sim \mathcal{N}(\mu_j, \sigma_j^2)$.*

Proof. Sketch: The performance of the team A increases linearly by $l_{\alpha,\beta}(\mu_\alpha + \mu_\beta)$ when a training instance is allocated to $\{a_\alpha, a_\beta\}$. Thus, allocating all training instances to $\{a_i, a_j\}$ provides the most performance improvement.

4.2 Allocating Training Instances

We consider two algorithms from the multi-armed bandit problem: the Upper Confidence Bound algorithm [2] and the Thompson sampling algorithm [9]. Next, we contribute an algorithm that solves the learning agents team formation problem by approximating the optimal allocation.

Algorithms from the Bandit Problem The computation of the UCB in the learning agents team formation problem is slightly different from the UCB algorithm in the multi-armed bandit problem, because the coordination of learning agent pairs change as training instances are allocated to them, while the probabilities are constant in the multi-armed bandit problem. Since the optimal solution is to always allocate all training instances to a single learning agent pair, the modified UCB estimates the UCB of a learning agent pair if all remaining training instances were allocated to it. The pair with the highest UCB is then allocated the training instance.

In the modified Thompson sampling (TS) algorithm, the estimate of the final coordination of a learning agent pair is computed by summing the estimate of its current coordination $\hat{\phi}_{i,j}$ and the estimate of the learning rate $\hat{l}_{i,j}$. The modified TS then retrieves a sample from the distribution and the learning agent pair with the highest sample is trained.

Learning Agents Team Formation Algorithm Algorithm 1 shows the pseudocode for approximating the optimal solution to the learning agents team formation problem. For each learning agent pair, the algorithm computes the maximum coordination of the pair, using the upper confidence bound of the coordination and learning rate estimates, assuming all remaining training instances are allocated to it (Line 4), i.e., summing the mean and standard deviation of the current estimates of $\phi_{i,j}^{(k)}$ and $l_{i,j}$. For all other learning pairs, the mean of the current estimates of the coordination are used (Line 5). The best possible team of size n^* with such an arrangement is found (Lines 6–9), and the corresponding learning agent pair is allocated the training instance (Line 10). The Kalman filter is updated using the observation (Line 11). The computational complexity of ApproxOptimalLinear is $\mathcal{O}(K|\mathcal{L}|\binom{N}{n^*}{n^*}^2)$, where the $\binom{N}{n^*}{n^*}^2$ term comes from Line 6, and could be reduced to ${n^*}^2$ by using the Synergy Graph team formation algorithm that approximates the optimal team [4], instead of finding the optimal team. We show the pseudo-code as such to ensure optimality, but if runtime is a concern then the approximation can be used.

The key difference between ApproxOptimalLinear and UCB is that in the latter, the pair $\{a_i, a_j\}$ with the highest $\widetilde{\phi}_{i,j}^{(K)}$ (as computed in Line 4 of Algo. 1)

Algorithm 1 Estimate learning rates to approximate the optimal allocation of training instances

```
ApproxOptimalLinear($K$)
 1: for $k = 1, \dots, K$ do
 2:     $(A_{\text{best}}, v_{\text{best}}) \leftarrow (\emptyset, -\infty)$
 3:     for all $\{a_i, a_j\} \in \mathcal{L}$ do
 4:         $\widetilde{\phi}^{(K)}_{i,j} \leftarrow (E(\hat{\phi}^{(k)}_{i,j}) + \sqrt{var(\hat{\phi}^{(k)}_{i,j})}) + (K - k + 1) \cdot (E(\hat{l}_{i,j}) + \sqrt{var(\hat{l}_{i,j})})$
 5:         $\widetilde{\phi}^{(K)}_{\alpha,\beta} \leftarrow E(\hat{\phi}^{(k)}_{\alpha,\beta}) \forall \{\alpha, \beta\} \neq \{i, j\}$
 6:         $A_{i,j} \leftarrow \text{argmax}_{(A \subseteq \mathcal{A} \text{ s.t. } |A| = n^*)} E(\widetilde{P}(A))$
 7:         if $E(\widetilde{P}(A_{i,j})) \geq v_{\text{best}}$ then
 8:             $\text{pair}_{\text{best}} \leftarrow \{a_i, a_j\}$
 9:             $(A_{\text{best}}, v_{\text{best}}) \leftarrow (A, E(\widetilde{P}(A_{i,j})))$
10:     $o_k \leftarrow \text{Train}(\text{pair}_{\text{best}})$
11:     $\text{KalmanUpdate}(\text{pair}_{\text{best}}, o_k)$
12: return $A_{\text{best}}$
```

would be allocated the training instance. In our algorithm, the upper confidence bound of a learning agent pair's coordination is used to estimate the performance of teams, and the training instance is allocated to the corresponding learning agent pair whose team has the highest estimated performance. Hence, the training instance is not always allocated to the learning agent pair with the highest upper confidence bound.

5 Agents that Improve Geometrically

In the previous section, we considered learning agent pairs whose coordination increased linearly with each training instance. However, a linear improvement may not reflect learning in many situations — typically there is a large amount of improvements in the early stages, but the marginal improvement decreases as more learning is done. In this section, we consider learning agent pairs that have a geometric improvement in coordination.

Specifically, we consider $\phi^{(k)}_{i,j} = \phi^{(0)}_{i,j} + F_{i,j}(k_{i,j}, l_{i,j})$ where the coordination gain function $F_{i,j}(k_{i,j}, l_{i,j}) = \sum_{k=1}^{k_{i,j}} l_{i,j} \cdot \gamma^{k-1}_{i,j}$. The coordination function is non-linear, and with each training instance, the learning agent pair's coordination is increased by a marginally smaller amount, i.e., the coordination gain between the k^{th} and $(k+1)^{\text{th}}$ training instance has a factor of $0 < \gamma_{i,j} < 1$ difference. Such a formulation brings about a new feature to the problem: if $|\mathcal{L}| \geq 2$, it is not always optimal to always train the same learning agent pair since a different pair may provide a higher increase in team performance. We are interested to consider how well the algorithms in the previous section will perform with such geometric improvements in coordination.

While the learning agents improve geometrically, the Kalman filter can still be used to estimate the learning rates, by assuming that the decay rate $\gamma_{i,j}$ is known, so only $l_{i,j}$ is unknown, similar to the linear case.

First, we explain how the algorithms of the previous section are modified to fit the new problem. Second, we consider the optimal allocation of training instances assuming all learning rates $l_{i,j}$ and decay factors $\gamma_{i,j}$ are known, and third, we contribute an algorithm that solves the learning agents team formation problem with geometric learning using the optimal solution as a guide.

5.1 Applying the Linear Algorithms to Agents with Geometric Improvements

The algorithm we contributed in the previous section, ApproxOptimalLinear (Algorithm 1), was designed for learning agent pairs that improved their coordination linearly. However, the algorithm only needs to be modified slightly to apply to learning agent pairs that improve their coordination geometrically.

Line 4 of Algo. 1 computes the estimated upper confidence bound of the learning agent pair's coordination assuming all remaining training instances are allocated to the pair. For geometric coordination improvements, the upper confidence bound of the coordination should be: $\widetilde{\phi}_{i,j}^{(K)} \leftarrow (E(\hat{\phi}_{i,j}^{(k)}) + \sqrt{var(\hat{\phi}_{i,j}^{(k)})}) + (E(\hat{l}_{i,j}) + \sqrt{var(\hat{l}_{i,j})}) \cdot \sum_{k'=1}^{K-\alpha+1} \gamma_{i,j}^{k'-1}$

Similar modifications also apply to the UCB and TS. These changes allow the algorithms to compute the final estimated coordination of the learning agent pairs when the improvements are geometric, while preserving the nature of the algorithms. We compare the performance of these algorithms in the geometric case in the experiments.

5.2 Optimally Allocating Training Instances

The changes to the algorithms described above provide an allocation of training instances in the geometric case, but in their computations, they assume that a learning agent pair is allocated the remainder of the training instances. In this subsection, we analyze what the optimal allocation of training instances should be in the geometric case.

We first consider the optimal allocation of training instances, given a fixed team A of size n^*.

The coordination gain $F_{i,j}(k_{i,j}, l_{i,j}) = \sum_{k=1}^{k_{i,j}} l_{i,j} \cdot \gamma_{i,j}^{k-1}$, which can be simplified to $F_{i,j}(k_{i,j}, l_{i,j}) = l_{i,j} \cdot \frac{1-\gamma_{i,j}^{k_{i,j}}}{1-\gamma_{i,j}}$.

The performance of A is:

$$P(A) = \frac{1}{\binom{|A|}{2}} \sum_{\{a_i,a_j\} \in A} (\phi_{i,j}^{(0)} C_{i,j} + l_{i,j} \cdot \frac{1-\gamma_{i,j}^{k_{i,j}}}{1-\gamma_{i,j}} C_{i,j})$$

Since the only variable is $k_{i,j}$ (the allocation of training instances), the optimal allocation Allocation(A, K) given A is: argmax $\sum_{\{a_i,a_j\}\in A^2\cap\mathcal{L}} l_{i,j}\cdot\frac{1-\gamma_{i,j}^{k_{i,j}}}{1-\gamma_{i,j}}C_{i,j}$ such that $\sum_{\{a_i,a_j\}\in A^2\cap\mathcal{L}} k_{i,j} = K$.

With such a formulation, the optimal allocation of training instances given K can be found using a non-linear integer solver. When A is unknown, the optimal allocation is:

$$\text{OptGeometric}(K) = \text{argmax}_{A\subseteq\mathcal{A}\text{ s.t. }|A|=n^*}\text{Allocation}(A, K)$$

However, computing the optimal allocation is infeasible given that the non-linear integer solver is run on every possible team, thus having a runtime of $\mathcal{O}(\binom{N}{n^*}2^{|\mathcal{L}|})$.

5.3 Allocating Training Instances to Agents that Improve Geometrically

We described the optimal allocation of training instances in the geometric case, that requires *a priori* knowledge of the learning rates $l_{i,j}$. Algorithm 2 uses the approach of the optimal solution to allocate the training instances.

Algorithm 2 Approximate the optimal solution with geometric learning rates

```
ApproxOptimalGeometric($K$)
1: for $k = 1, \ldots, K$ do
2:     $(A_{\text{best}}, v_{\text{best}}) \leftarrow (\emptyset, -\infty)$
3:     for all $A \subseteq \mathcal{A}$ s.t. $|A| = n^*$ do
4:         $(v_A, k'_k, \ldots, k'_K) \leftarrow$ Allocation'$(A, K - k + 1)$
5:         if $v_A \geq v_{\text{best}}$ then
6:             $\text{pair}_{\text{best}} \leftarrow k'_k$
7:             $(A_{\text{best}}, v_{\text{best}}) \leftarrow (A, v_A)$
8:     $o_k \leftarrow \text{Train}(\text{pair}_{\text{best}})$
9:     $\text{KalmanUpdate}(\text{pair}_{\text{best}}, o_k)$
10: return $A_{\text{best}}$
```

The function Allocation' (Line 4 of Algo. 2) uses a non-linear integer solver to allocate the remaining training instances, and is similar to the Allocation function of the optimal solution, except that the upper confidence bound of the learning rates are used, i.e., $E(\hat{l}_{i,j}) + \sqrt{var(\hat{l}_{i,j})}$, since $l_{i,j}$ is unknown and being estimated by the Kalman filter. Allocation' returns the best allocation returned by the integer solver $(k'_k, \ldots, k'_K)$, and the value of the final team (v_A) if the allocation is performed. The best allocation is sorted so that the learning agent pair with the highest contribution gain from the allocation is returned.

However, since the non-linear integer solver is run for all possible teams A, and for K iterations, the algorithm ApproxOptimalGeometric is infeasible when the number of possible teams is large, having a runtime of $\mathcal{O}(K\binom{N}{n^*}2^{|\mathcal{L}|})$. We present ApproxOptimalGeometric as a baseline to consider if computation was not an issue, while still preserving the nature of the learning agents team formation problem, i.e., that the learning rates of the learning agent pairs are initially unknown. We later show in our experiments that our ApproxOptimalLinear algorithm (Algo. 1) has similar performance with a much smaller runtime.

6 Experiments and Results

This section describes the experiments we conducted to compare the performance of ApproxOptimalLinear and ApproxOptimalGeometric against UCB and TS.

6.1 Experimental Setup

To generate the agent capabilities and pairwise coordination, we used a Synergy Graph [4] with weighted edges of $|\mathcal{A}| = 10$ vertices, with the agent capabilities $C_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ such that $\mu_i \in (\frac{m}{2}, \frac{3m}{2})$ and $\sigma_i^2 \in (0, m^2)$ where the multiplier $m = 100$. We used the Synergy Graph model as it provided a means to compute the coordination between agent pairs, and generated the agent capabilities with a large variation among the agents.

We varied $|\mathcal{L}|$, the number of learning agent pairs, to be between 5 and 9, and randomly selected the $|\mathcal{L}|$ learning agent pairs. We limited the size of $|\mathcal{L}|$ since the computation of the optimal allocation is exponential in the geometric case. For each learning agent pair, we randomly sampled their learning rates, such that in the linear case, the learning rate $l_{i,j} \in (0, 0.1)$, and in the geometric case, the initial learning rate $l_{i,j} \in (0, 1)$ and the decay rate $\gamma_{i,j} \in (0.9, 0.95)$. The bounds of the learning and decay rates were chosen such that the coordination gains after allocating the training instances do not completely overshadow the initial team performance before learning, and that the learning rates do not decay too quickly in the geometric case.

For each value of $|\mathcal{L}|$, we generated 50 Synergy Graphs for the linear case and 50 Synergy Graphs for the geometric case, and the corresponding variables for the learning agent pairs. We also varied the number of training instances, i.e., $K = 20, 40, \ldots, 280, 300$.

6.2 Evaluating the Algorithms

Figure 1 shows the performance of teams formed by the various algorithms when the learning agent pairs increase their coordination linearly, and when the number of learning agent pairs $|\mathcal{L}| = 5$. The results were similar for other values of $|\mathcal{L}|$, so we only present $|\mathcal{L}| = 5$. The dotted orange line shows the performance of the optimal allocation. ApproxOptimalLinear has a similar performance to both UCB and TS, showing that considering the compatibility of a learning agent pair is sufficient to solve the problem, without having to consider the overall performance of the team. We believe this is due to the linear setup, as shown from how the optimal allocation assigns all the training instances to a single pair.

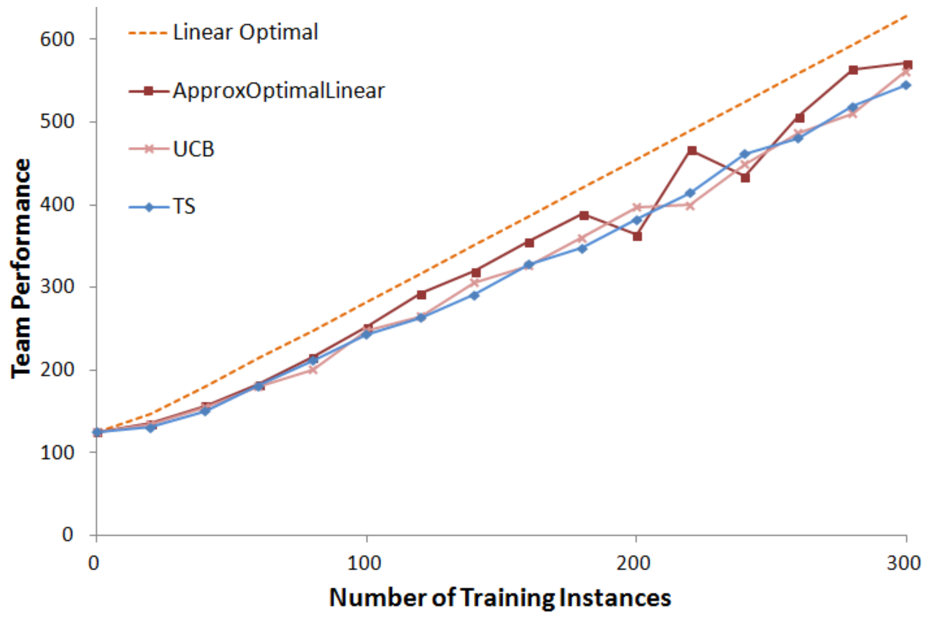

Fig. 1: The performance of teams formed with various algorithms when coordination increases linearly.

Further, we believe the geometric case, which we discuss next, is more realistic in terms of the improvements of learning agents.

When learning agent pairs increase their coordination geometrically, ApproxOptimalLinear significantly outperforms UCB and TS. Figure 2 shows the performance of the various algorithms. The optimal geometric allocation is shown with the dotted black line, and the optimal linear allocation is shown with the dotted orange line. ApproxOptimalLinear, UCB, and TS estimate the learning rates and allocated the training instances iteratively, while the optimal linear allocation knows the learning rates and computes a single learning agent pair that receives all the training instances. UCB and TS do not perform well in the geometric case, being close to the optimal linear allocation, even though the algorithms were updated for the geometric case (Section 5.1). Hence, our results show that it is important to keep the team formation goal in mind while allocating training instances in the geometric case.

The dark purple line shows the performance of ApproxOptimalGeometric. ApproxOptimalLinear performs slightly worse than ApproxOptimalGeometric. Hence, even though the optimal geometric solution and approximate geometric solution requires a non-linear integer solver, an allocation algorithm such as ApproxOptimalLinear is sufficient.

We repeated the experiments for $|\mathcal{L}| = 6, \ldots, 9$ and noted similar trends, and similarly when the other variables of our experiments were changed, e.g., the multiplier m used to generate agent capabilities. Thus, through our experiments,

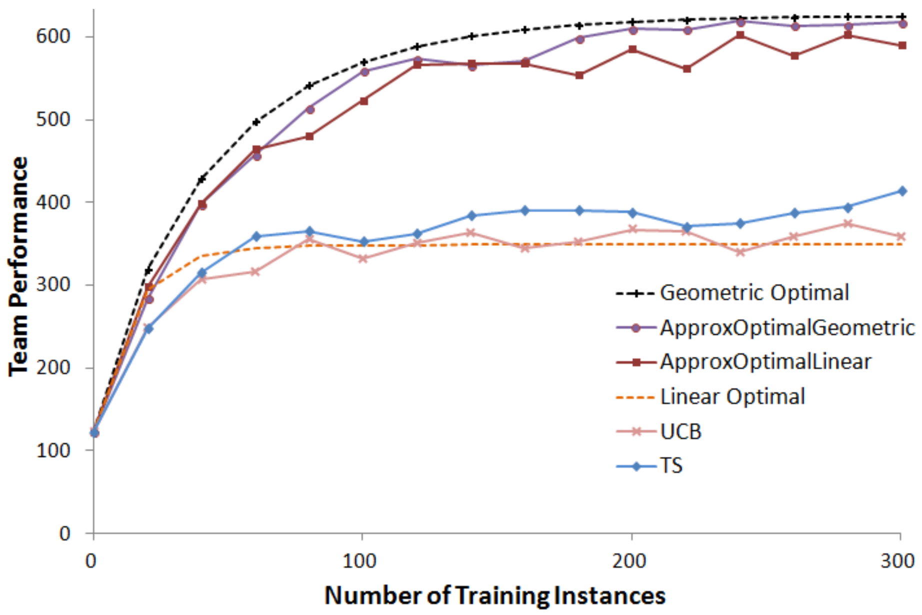

Fig. 2: The performance of teams formed with various algorithms when coordination increases geometrically.

we showed that our algorithm, ApproxOptimalLinear, is suitable for solving the learning agents problem in both the linear and geometric case, performing close to optimal in both scenarios. Further, the computational cost for our algorithm is polynomial even in the geometric case where the optimal solution is exponential.

7 Conclusion

We are interested in agents that learn to coordinate with their teammates, in particular, pairs of agents that improve their coordination through training instances. We formally defined the learning agents team formation problem, where the goal is to allocate a fixed number of training instances to learning agent pairs that have heterogeneous capabilities and learning rates, so as to form a team, i.e., select a subset of the agents, that maximizes the team performance.

We considered two categories of learning agent pairs, those that improved their coordination linearly, and those that improved geometrically, i.e., the marginal improvement decreased by a constant factor with each training instance. We defined the optimal allocation of training instances in both categories, assuming that the learning rates of the agent pairs are known. We contributed two algorithms to solve the learning agents team formation problem, one for the linear category and one for the geometric category, that estimate the learning rates and allocate the training instances.

There are similarities between the learning agents team formation and multi-armed bandit problems, and we compared our algorithms against the upper confidence bound and Thompson sampling algorithms from the multi-armed bandit problem. In extensive simulated experiments, we showed that our algorithms performed similarly to the bandit algorithms in the linear category, and outperformed them in the geometric category, finding near-optimal solutions in both categories, demonstrating the efficacy of our algorithms in the learning agents team formation problem.

Acknowledgments

This work was partially supported by the Air Force Research Laboratory under grant number FA87501020165, by the Office of Naval Research under grant number N00014-09-1-1031, and the Agency for Science, Technology, and Research (A*STAR), Singapore. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity. This work was supported by the A*STAR Computational Resource Centre through the use of its high performance computing facilities.

References

1. N. Agmon and P. Stone. Leading Ad Hoc Agents in Joint Action Settings with Multiple Teammates. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 341–348, 2012.
2. P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256, 2002.
3. S. Barrett, P. Stone, and S. Kraus. Empirical Evaluation of Ad Hoc Teamwork in the Pursuit Domain. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 567–574, 2011.
4. S. Liemhetcharat and M. Veloso. Modeling and Learning Synergy for Team Formation with Heterogeneous Agents. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 365–375, 2012.
5. S. Liemhetcharat and M. Veloso. Weighted Synergy Graphs for Role Assignment in Ad Hoc Heterogeneous Robot Teams. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5247–5254, 2012.
6. L. Panait and S. Luke. Cooperative Multi-Agent Learning: The State of the Art. *Journal of Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.
7. P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In *Proceedings of the International Conference on Artificial Intelligence*, 2010.
8. M. Tan. Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Proceedings of the International Conference on Machine Learning*, pages 330–337, 1993.
9. W. Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3/4):285–294, 1933.
10. K. Tuyls and A. Nowe. Evolutionary Game Theory and Multi-Agent Reinforcement Learning. *Journal of Knowledge Engineering Review*, 20(1):65–90, 2005.