# Modeling and Learning Synergy for Team Formation with Heterogeneous Agents

Somchaya Liemhetcharat
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
som@ri.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
veloso@cs.cmu.edu

## ABSTRACT

The performance of a team at a task depends critically on the composition of its members. There is a notion of synergy in human teams that represents how well teams work together, and we are interested in modeling synergy in multi-agent teams. We focus on the problem of team formation, i.e., selecting a subset of a group of agents in order to perform a task, where each agent has its own capabilities, and the performance of a team of agents depends on the individual agent capabilities as well as the synergistic effects among the agents. We formally define synergy and how it can be computed using a synergy graph, where the distance between two agents in the graph correlates with how well they work together. We contribute a learning algorithm that learns a synergy graph from observations of the performance of subsets of the agents, and show that our learning algorithm is capable of learning good synergy graphs without prior knowledge of the interactions of the agents or their capabilities. We also contribute an algorithm to solve the team formation problem using the learned synergy graph, and experimentally show that the team formed by our algorithm outperforms a competing algorithm.

## Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

## General Terms

Algorithms, Experimentation

## Keywords

Capability, synergy, team formation, heterogeneous

## 1. INTRODUCTION

It is clear that the performance of a team, in terms of the outcome of the task, depends on the team composition. The term *synergy* is commonly used in human teams, and describes how well the team works together. We extend this notion of synergy from human teams to multi-agent teams, and seek to model and quantify it for effective team formation at a task.

**Appears in:** *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.


Research in agent capabilities, which we describe in detail in the related work section, typically focus on single-agent capabilities, and not the interactions between multiple agents. Similarly, research in coalition formation typically does not seek to model how the value of a coalition is computed. In this paper, we introduce a model that captures synergy from the interactions among large groups of agents.

Concretely, we abstract the problem as finding the best subset of agents to complete a task, where each agent has its own set of capabilities. The performance of a team of agents at the task depends both on the capabilities of the agents, and the synergy among the members of the team. This model of performance among the agents is initially unknown, but observations of the performance of groups of agents can be made. From these observations, a model of synergy within the agents is learned, and the optimal subset of the agents is then selected for the task.

Formally, we define a synergy graph that models single-agent capabilities and the interactions among the agents. We then define pairwise synergy, i.e., how well a pair of agents will perform together, and define synergy in groups of agents. We then contribute an algorithm to find the optimal team to perform the task from a synergy graph, and an algorithm that learns a synergy graph from observations of interactions among small groups of agents. In our experiments, we show that the learned synergy graph matches closely to the hidden model that generated the observations, and the team formed by searching this learned synergy graph performs very well. In addition, we use a probabilistic robot capability model introduced in [8], and show that the synergy graph learned from observations of this probabilistic model leads to the formation of a team that outperforms the team selected by the ASyMTRe algorithm [8].

The format of our paper is as follows: in Sec. 2, we discuss related work and the differences with our work. In Sec. 3, we formally define the problem, and contribute our synergy graph model and the algorithm for team formation. In Sec. 4, we contribute an algorithm that learns a synergy graph based on observations of the performance of agents at the task. In Sec. 5, we describe our experiments and results, and we summarize our contributions in Sec. 6.

## 2. RELATED WORK

In heterogeneous teams, agents and robots have different capabilities. There has been a large amount of research in the task-allocation and role assignment domains [3], but the capabilities are typically binary, i.e., whether a robot is capable of performing a sub-task is usually due to its physical characteristics such as the presence of an arm. There has

also been some research in modeling capabilities as values, where higher values indicate better task performance [5], and with a Normal distribution to represent the uncertainty in the agents' performance [4]. However, while modeling capabilities of single agents and robots has been extensively studied, there has been limited work in modeling the capabilities of teams of agents, other than a sum of their individual capabilities, or a binary to represent whether the team can perform a sub-task. Pairwise capabilities between agents has been studied [7], and the coupling of robot capabilities to perform complex tasks using schemas with the ASyMTre algorithm [8], which we elaborate further below.

In the ASyMTRe algorithm, robot capabilities are modeled with schemas, that define inputs and outputs of information types (e.g., the global position of the robot) [8]. Each robot has a set of schemas with probabilities of success. The task is defined as a set of desired outputs, and a multi-robot team is formed to complete the task by creating a joint plan of the robots' schemas. Teams are ranked by a utility function that balances the probability of success and the cost of execution. ASyMTRe is an anytime algorithm that requires prior knowledge of the agent's capabilities and probabilities of success in order to rank potential teams, while our approach does not need such *a priori* information. Our approach models and learns the interactions between large groups of agents to form an effective team, and we compare the performance of our algorithm against ASyMTRe.

Coalition formation is a related field, where every possible subset of agents is given a value, and the goal is to partition the agents so as to maximize the sum of values. However, most of the research in the field has focused on how to achieve the best partitioning [9]. There has been some recent work in modeling how the value of a coalition is affected by externalities [1], but not how the value of a coalition is derived based on the composition of its members. Service and Adams recently applied coalition formation to solve task allocation, where the goal is to maximize the utility gained from completing tasks, and taking into account the resources/services that the agents can provide [10]. However, they use a market-based approach to solve the task allocation and do not model the synergistic effects across agents that have the same service. Our approach models varying quality in the capabilities of agents and how interactions in a team can amplify the results.

In the social network domain, there has been much research in selecting teams based on the interactions of agents in the social network graph. Lappas, Liu and Terzi studied how to find a team of experts that accomplish a goal while minimizing the communication cost in a social network graph [6]. Similarly, Dorn and Dustdar studied how to compose near-optimal expert teams by trading-off between coverage and communication cost [2]. We extend the use of a social network graph to model synergy between teams of agents, where the distance in the graph correlates with how well agents work together, and not the communication cost between the agents, and thus, the task performance is directly affected by the structure of the graph. Further, we contribute an algorithm to learn the structure of the graph from observations.

# 3. MODELING SYNERGY FOR TEAM FORMATION

Let $\mathcal{A}$ be a set of agents, and the task be $T$. Let the task $T$ be divided into $M$ independent sub-tasks, and let $F : 2^{\mathcal{A}} \rightarrow X$, where $X$ is a $M$-dimensional random variable with unknown distribution. The function $F$ represents the "world", so $F$ is unknown but samples of $F(A)$ can be retrieved for agent teams $A \subseteq \mathcal{A}$, where the size of a team can vary from 1 to $|\mathcal{A}|$. A sample of $F(A)$ corresponds to the values attained by $A$ at the $M$ sub-tasks.

Let $V : \mathbb{R}^M \rightarrow \mathbb{R}$ be a value function that computes the overall value at the task $T$ based on the values of the $M$ sub-tasks. Examples of $V$ are: $V_{sum}(X) = \sum_{m=1}^{M} X(m)$, $V_{max}(X) = \max_{m=1}^{M} X(m)$, and $V_{task}(X) = V_{sum}(X)$ iff $X(m) > \tau \, \forall \, m \in [1, M]$, and 0 otherwise, where $X(m)$ is the $m^{th}$ component of $X$. $V_{sum}$ and $V_{max}$ are the summation and maximum functions, while $V_{task}$ is a composite function that returns 0 if the performance of any sub-task is below a threshold $\tau$, and is the summation otherwise.

The goal is to find a team of agents $A^* \subseteq \mathcal{A}$ such that $\forall \; A \subseteq \mathcal{A}, \; V(F(A^*)) \geq V(F(A))$, i.e., $A^*$ receives the highest value at the task $T$.

## 3.1 Modeling Task-Based Relationships

In order to solve this problem of forming the optimal team to complete the task $T$, we create a model of $F$ based on samples of $F(A)$. We assume that there is some task-based relationship between the agents that we can model. Research in the social network field use social graphs and communication costs to form effective teams [6, 2]; we model task-based relationships with a task-based network between the agents. As such, we form a task-based graph, where vertices are agents, and the edges represent the task-based relationship between the agents.

One method to model this relationship is with a fully-connected graph, where the weights of the edges represent how well agents work together (smaller numbers mean agents work better together, i.e., with lower cost). For example, Fig. 1 shows a 3-agent graph, where $a_1$ and $a_2$ work well together compared to other pairs.

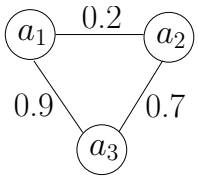


**Figure 1: A fully-connected task-based graph where the task-based relationship between agents are represented by edge weights.**

However, a fully-connected graph does not capture transitivity in the task-based relationship. For example, transitivity would mean that if $a_1$ works very well with $a_2$, and $a_2$ works very well with $a_3$, then $a_1$ should work well with $a_3$. To capture transitivity in the task-based relationship, we can use a connected graph (instead of a fully-connected one), where the minimum distance between agents in the graph is inversely correlated with their task-based relationship. Fig. 2 shows a modification from Fig. 1 where the edge $\{a_1, a_3\}$ has been removed, as an example that still preserves the distance between the agents.

In order to model the inverse correlation between the distance of agents and their task-based relationship, we introduce a weight function $w : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, where $w(d(a_1, a_2))$ returns the task-based value of agents $a_1$ and $a_2$ based on

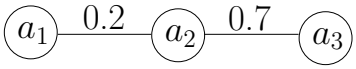


**Figure 2: A connected task-based graph where the task-based relationship between agents is a function of the shortest distance between them.**

the minimum distance between them in the graph (using the function $d$). Further, $w(d)$ is always positive and monotonically decreases as $d$ increases. Intuitive examples of $w$ are $w(d) = \frac{1}{d}$, and $w(d) = \exp(-\frac{d \ln 2}{h})$, which is an exponential decay function with half-life $h$.

As a simplifying assumption, we assume the edges in the task-based graph are unweighted (all edges have weight 1), and the weight function is still capable of fully capturing the task-based relationship.

## 3.2 Quantifying Performance at the Task

We want to model $F$ based on samples of $F(A)$, and so far we have introduced a graphical model to capture the task-based relationships between agents. However, there is an innate capability of agents that is still unmodeled. For example, even if $a_1$ works equally well with $a_2$ and with $a_3$, the value $F(\{a_1, a_2\})$ may be consistently higher than $F(\{a_1, a_3\})$, if $a_2$ is "better" at the task than $a_3$.

As such, the graph structure alone is insufficient to completely model $F$. We thus add a value to each vertex. Although $F$ returns an unknown distribution, we assume that it can be represented by $M$ Normal distributions, where each Normal distribution models the agent's performance at a sub-task. Fig. 3 shows a 6-agent graph, where $M = 1$. We use Normal distributions because the single peak corresponds to the agent's average performance, and the symmetric spread corresponds to the agent's variability.

Now, we formally define a synergy graph:

*Definition 1.* A **synergy graph** $S$ is a tuple $\{G_S, N_S\}$, such that $G_S = (V_S, E_S)$ is a connected graph, and $N_S$ is set of Normal distributions, where:

- $v_a \in V_S$ is a vertex in $G_S$ and represents an agent $a \in \mathcal{A}$,
- $E_S$ are unweighted edges in $G_S$, and
- $N_a = (N_{a,1}, \ldots, N_{a,M}) \in N_S$ is a list of $M$ Normal distributions, where $N_{a,m} \sim \mathcal{N}(\mu_{a,m}, \sigma^2_{a,m})$ is the capability of agent $a$ at sub-task $m \in [1, M]$.

Using a synergy graph, we can compute the performance of a pair of agents:

*Definition 2.* The **pairwise synergy** $\mathbb{S}_2(a, a')$ between 2 agents $a, a' \in \mathcal{A}$ in a synergy graph $S$ is a list of $M$ Normal distributions, given by $w(d(v_a, v_{a'})) \cdot (N_a + N_{a'})$, where each component of $N_a$ and $N_{a'}$ is summed independently, $d(v_a, v_{a'})$ is the shortest distance between the $v_a, v_{a'}$ in $G_S$, and $w(d)$ is a positive weight function that monotonically decreases as $d$ increases.

Using this definition of synergy between a pair of agents, we define synergy within a group of agents, i.e., their task performance, below:

*Definition 3.* The **synergy** $\mathbb{S}(A)$ of a set of agents $A \subseteq \mathcal{A}$ in a synergy graph $S$ is the average of the pairwise synergy of its components, i.e., $\frac{1}{\binom{|A|}{2}} \cdot \sum_{\{a,a'\} \in A} \mathbb{S}_2(a, a')$

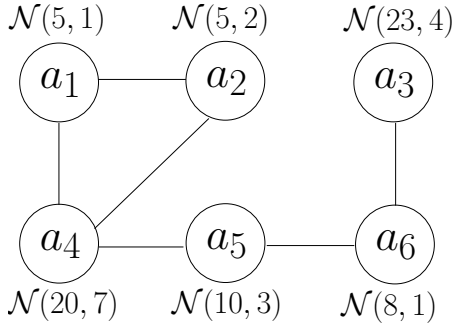


**Figure 3: A synergy graph with 6 agents. Each vertex represents an agent, and the distance between vertices in the graph indicate how well agents work together. Agents have lists of Normal distributions that correspond to their capabilities at the $M$ sub-tasks. In this example, $M = 1$.**

Thus, $\mathbb{S}(A)$ returns $\{\mathcal{N}(\mu_{A,1}, \sigma^2_{A,1}), \ldots, \mathcal{N}(\mu_{A,M}, \sigma^2_{A,M})\}$, a list of $M$ Normal distributions, indicating the task performance of the team $A \subseteq \mathcal{A}$. From Defs. 2 and 3:

$$\mu_{A,m} = \frac{1}{\binom{|A|}{2}} \sum_{\{a,a'\} \in A} w_{a,a'} \cdot (\mu_{a,m} + \mu_{a',m}) \quad (1)$$

$$\sigma^2_{A,m} = \frac{1}{\binom{|A|}{2}^2} \sum_{\{a,a'\} \in A} w^2_{a,a'} \cdot (\sigma^2_{a,m} + \sigma^2_{a',m}) \quad (2)$$

where $w_{a,a'} = w(d(v_a, v_{a'}))$ and $N_{a,m} \sim \mathcal{N}(\mu_{a,m}, \sigma^2_{a,m})$ are the agent capabilities at sub-task $m \in [1, M]$, assumed to be independent.

While the definition of synergy involves the summation of individual capabilities, it is weighted by the distance of agents in the synergy graph, and as such, the addition or removal of specific agents can have a large impact on the total score of a team. For example, from Fig. 3, suppose that $w(d) = \frac{1}{d}$. Then, the team $\{a_1, a_2\}$ has a mean score of 10, but the addition of $a_3$ lowers the mean to 8. Conversely, the team $\{a_1, a_2, a_4\}$ increases the mean to 20, even though the individual capability of $a_4$ is lower than $a_3$.

We defer the learning of synergy graphs from samples of $F$ to a later section, and first describe how to find the best team $A^* \subseteq A$ given a synergy graph $S$.

## 3.3 Composing an Effective Team

In this section, we introduce an algorithm to approximate the optimal team composition for the task, in terms of the task performance, given a synergy graph. The goal is to find the optimal team $A^* \subseteq \mathcal{A}$ from a synergy graph $S$. We assume that the size of the optimal team (i.e., $n^* = |A^*|$) is known. This is a reasonable assumption, since the size of teams are typically limited by external factors, e.g., a cost budget, size restrictions. For example, the size of teams in sports is fixed, and also in tasks that require handing of a fixed number of devices, such as the operators of an ambulance. In addition, if $n^*$ is unknown, then our approach can be run iteratively for increasing $n$, and then return the optimal team found across all $n$.

$\mathbb{S}$ calculates the list of $M$ Normal distributions of the team's performance. In order to rank teams, each Normal distribution needs to be converted into a single number. To do so, we use the evaluation function introduced by us in [7], that balances the mean and variance of a Normal distribu-

**Algorithm 1** Approximating the optimal team of size $n$

ApproxOptimalTeam($S, n, \rho$)
1: $A \leftarrow$ `GenerateRandomTeam`$(\mathcal{A}, n)$
2: $\{N_{A,1}, \ldots, N_{A,M}\} \leftarrow \mathbb{S}(A)$
3: $v \leftarrow V($`Evaluate`$(N_{A,1}, \rho), \ldots,$ `Evaluate`$(N_{A,M}, \rho))$
4: **for** $k = 1$ to $k_{max}$ **do**
5: $\quad A' \leftarrow$ `RandomTeamNeighbor`$(A)$
6: $\quad \{N_{A',1}, \ldots, N_{A',M}\} \leftarrow \mathbb{S}(A')$
7: $\quad v' \leftarrow V($`Evaluate`$(N_{A',1}, \rho), \ldots,$ `Evaluate`$(N_{A',M}, \rho))$
8: $\quad$ **if** `P`$(v, v',$ `Temp`$(k, k_{max})) >$ `random`() **then**
9: $\quad\quad A \leftarrow A'$
10: $\quad\quad v \leftarrow v'$
11: **return** $A$

tion using a risk factor $\rho \in (0, 1)$, such that when $\rho > \frac{1}{2}$, distributions with higher variances attain higher values. As such, `Evaluate`$(N_{A,m}, \rho) = \mu_{A,m} + \sigma_{A,m} \cdot \Phi^{-1}(\rho)$, where $\Phi^{-1}$ is the inverse standard Normal cumulative distribution function. Thus, the $M$ Normal distributions are converted into $M$ real numbers, and the value function $V : \mathbb{R}^M \rightarrow \mathbb{R}$ is computes the final task value.

Algo. 1 finds an approximation of the optimal team of a given size $n$. A random team is first generated, where $n$ agents are randomly chosen. Next, simulated annealing is performed to optimize the team configuration, where a neighbor of the current team is created by replacing one agent with another agent not currently in the team. Lines 4-10 of Algo. 1 implements simulated annealing, where `Temp` is a temperature schedule, `P`$(v, v', t)$ returns a value between 0 and 1 given values $v, v'$ and a temperature $t$.

Our team formation algorithm runs in $O(|A|)$ time if $n^*$ is known. Otherwise, Algo. 1 is run for increasing $n$ for a total runtime of $O(|A|^2)$. A brute-force algorithm would take $O(\binom{|A|}{n^*})$ if $n^*$ is known, and $O(2^{|A|})$ otherwise.

# 4. LEARNING SYNERGY GRAPHS

The previous section formalizes synergy and defines a synergy graph, which captures how members of a heterogeneous team interact. In addition, an algorithm to find the optimal team for a task was presented. However, in order to quantify synergy within a team, we first need to learn a synergy graph. In this section, we contribute an algorithm that learns a synergy graph from observations of task performance of groups of agents.

The value function $F : 2^{\mathcal{A}} \rightarrow X$ is unknown, but samples of $F(A)$ can be obtained for $A \subseteq \mathcal{A}$. $F$ can be likened to a black-box system, or the real world where the teams, i.e., $A \subseteq \mathcal{A}$, perform the task and the value of their performance at each of the $M$ sub-tasks is observed.

*Definition 4.* An **observation** $o_A$ is a list of $M$ values corresponding to an observed overall performance of members $A \subseteq \mathcal{A}$ at the $M$ sub-tasks, i.e., $o_A = F(A)$ for one sample of $F$. An **observation group** $O_A = \bigcup o_A$ is the set of all observations of $A$ at the task.

Each observation group $O_A$ contains all the observations of a unique group $A$. We assume that $\mathcal{A}$, the set of all agents, is known *a priori*. Otherwise, $\mathcal{A} = \bigcup_{O_A} A$. From these observation groups $O_A$, we define the observation set:

*Definition 5.* An **observation set** $O$ is the set of all observation groups, i.e., $O = \bigcup \{O_A\}$.

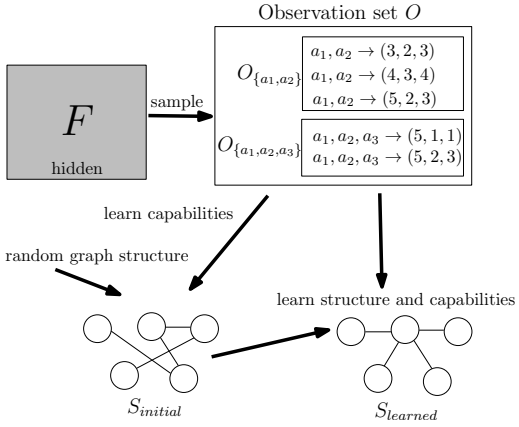


**Figure 4: The process of learning from observations. Each observation is a list of $M$ numbers, corresponding to each sub-task. The individual capabilities of agents in the synergy graphs are not shown.**

**Algorithm 2** Create a Synergy Graph from observations

CreateSynergyGraph($O, \mathcal{A}$)
1: $G \leftarrow$ `GenerateRandomGraph`$(\mathcal{A})$
2: $N \leftarrow$ `EstimateCapability`$(O, G)$
3: $S_{initial} \leftarrow \{G, N\}$
4: $S_{learned} \leftarrow S_{initial}$
5: $l \leftarrow$ `CalculateLogLikelihood`$(O, S)$
6: **for** $k = 1$ to $k_{max}$ **do**
7: $\quad G' \leftarrow$ `RandomNeighbor`$(G)$
8: $\quad N' \leftarrow$ `EstimateCapability`$(O, G')$
9: $\quad S' \leftarrow \{G', N'\}$
10: $\quad l' \leftarrow$ `CalculateLogLikelihood`$(O, S')$
11: $\quad$ **if** `P`$(l, l',$ `Temp`$(k, k_{max})) >$ `random`() **then**
12: $\quad\quad S_{learned} \leftarrow S'$
13: $\quad\quad l \leftarrow l'$
14: **return** $S_{learned}$

Fig. 4 illustrates the process of learning a synergy graph. The function $F$ is sampled to obtain observations, which form the observation set $O$. An initial synergy graph $S_{initial}$ is created from a random graph structure and learned capabilities from the observation set. Subsequently an iterative algorithm is used to learn the synergy graph structure that best fits the observation set.

Algo. 2 explains this learning process in detail. First, `GenerateRandomGraph` creates a random graph $G$ from agents $\mathcal{A}$, such that $G$ is a connected graph, i.e., all vertices are connected through chains of edges. Next, `EstimateCapability` estimates the individual agent capabilities $N$, using the observation set $O$ and graph $G$. The initial synergy graph $S$ is then created from $G$ and $N$, and the log-likelihood of the observations given $S$ is calculated using `CalculateLogLikelihood`. Simulated annealing is then performed to converge on a synergy graph, where `P`$(l, l', t)$ returns a value between 0 and 1 given values log-likelihoods $l, l'$ and a temperature $t$. The log-likelihood of the observation set given a synergy graph is used as the maximizing criterion for simulated annealing, as the goal is to find a synergy graph that best matches the observations, i.e., is most likely to have produced the observations given its graph structure and individual capabilities. `RandomNeighbor`$(G)$ is a function that takes an existing graph $G$, and either adds a new random

edge between two vertices, or removes an existing edge subject to the constraint that $G$ remains a connected graph.

Algo. 3 estimates the individual agent capabilities, using the observation set $O$ and graph $G$. Matrices $\mathcal{M}$ and $b$ are created such that $\mathcal{M}x = b$, e.g., $\mathcal{M}_\mu\ x_\mu = b_\mu$, where $x_\mu = [\mu_{a_1}, \ldots, \mu_{a_{|\mathcal{A}|}}]^T$. Each row in $\mathcal{M}$ and $b$ corresponds to an observation group $O_A$ in $O$, using Eqns. 1 and 2. Each column in $\mathcal{M}$ corresponds to an agent in $\mathcal{A}$. A least squares solver is run to find $x$, which corresponds to the means and variances of the agent's capabilities at the $m^{th}$ sub-task.

For example, suppose that $M = 1$ and $O_{\{a_1,a_2\}} = \{3, 4, 5\}$, i.e., the team $\{a_1, a_2\}$ was sampled 3 times using $F$ and received value 3 for the first sample, 4 for the second, and 5 for the third. This observation group would form a row $[\alpha, \alpha, 0, \ldots]$ in $\mathcal{M}_\mu$, and a row $[\alpha^2, \alpha^2, 0, \ldots]$ in $\mathcal{M}_{\sigma^2}$, where $\alpha = w(d(v_{a_1}, v_{a_2}))$. $b_\mu$ and $b_{\sigma^2}$ would then have a row with values 4 (the mean of the 3 observations) and 1 (the variance) respectively. Thus, each observation group creates a row in the $\mathcal{M}$ and $b$ matrices, and a least-squares solver is used to solve for the means and variances of each agent.

`CalculateLogLikelihood` computed the log-likelihood of the observations, given a synergy graph $S$. In order to do so, for each observation group $O_A$ in $O$, the synergy $\mathcal{N}(\mu_{A,m}, \sigma^2_{A,m})$ of the group $A \subseteq \mathcal{A}$ at the $m^{th}$ sub-task is calculated using $\mathbb{S}$. The log-likelihood of each observed value in the observation $O_A$ is then computed, and summed across all the observations and sub-tasks.

Overall, learning a synergy graph (Algo. 2) takes $O(r^3)$ time, where $r$ is the number of observation groups, due to the least-squares solver in Algo. 3.

**Algorithm 3** Estimate the individual agent capabilities

EstimateCapability$(O, G)$

Let $\mathcal{A} = \{a_1, \ldots, a_{|A|}\}$.
Let $G = (V_G, E_G)$, and $V_G = \{v_{a_j} : a_j \in \mathcal{A}\}$.
Let $O = \{O_{A_1}, \ldots, O_{A_r}\}$, where $A_i \subseteq \mathcal{A}$.
$\mathcal{M}_\mu \leftarrow \mathbf{0}_{r\times|\mathcal{A}|}$
$\mathcal{M}_{\sigma^2} \leftarrow \mathbf{0}_{r\times|\mathcal{A}|}$
$b_\mu \leftarrow \mathbf{0}_{r\times 1}$
$b_{\sigma^2} \leftarrow \mathbf{0}_{r\times 1}$
**for** $m = 1, \ldots, M$ **do**
  **for all** $O_{A_i} \in O$ **do**
    **for all** $a_j \in A_i$ **do**
      $\mathcal{M}_\mu(i,j) \leftarrow \frac{1}{\binom{|A_i|}{2}} \sum_{\{a_j,a\}\in A_i} w(d(v_{a_j}, v_a))$
      $\mathcal{M}_{\sigma^2}(i,j) \leftarrow \frac{1}{\binom{|A_i|}{2}^2} \sum_{\{a_j,a\}\in A_i} w(d(v_{a_j}, v_a))^2$
    // $o(m)$ is the $m^{th}$ component of observation $o$
    $b_\mu(i) \leftarrow \frac{1}{|O_{A_i}|} \sum_{o\in O_{A_i}} o(m)$
    $b_{\sigma^2}(i) \leftarrow \frac{1}{|O_{A_i}|-1} \sum_{o\in O_{A_i}} (o(m) - b_\mu(i))^2$
  $means \leftarrow$ `LeastSquares`$(\mathcal{M}_\mu, b_\mu)$
  $variances \leftarrow$ `LeastSquares`$(\mathcal{M}_{\sigma^2}, b_{\sigma^2})$
  **for all** $a_j \in \mathcal{A}$ **do**
    $N_{a_j,m} \sim \mathcal{N}(means(j), variances(j))$
$N \leftarrow \{\}$
**for all** $a_j \in \mathcal{A}$ **do**
  $N \leftarrow N \cup \{(N_{a_j,1}, \ldots, N_{a_j,M})\}$
**return** $N$

# 5. EXPERIMENTS AND RESULTS

In order to test the efficacy of our synergy graph model, the learning algorithm to create synergy graphs from observations, and the performance of teams formed from the learned models, we split our experiments into two phases.

In the first phase, the hidden function $F$ is set to be a synergy graph model, and our experiments are designed to show that the learned synergy graph matches the hidden synergy graph (in $F$) well. Further, we show that the teams formed from the learned synergy graph performs effectively compared to both the team formed from the hidden synergy graph, as well as the optimal team found by brute force.

In the second phase of our experiments, we compare our algorithms to the ASyMTRe algorithm [8]. The hidden function $F$ follows the probabilistic model of robot capabilities in [8], i.e., $F$ does not return a Normal distribution, and we learn a synergy graph model from observations of teams' performances. We then show that the team found from our learned synergy graph outperforms that of ASyMTRe.

## 5.1 $F$ as a Synergy Graph

In order to create random synergy graphs for the function $F$, we first defined $|\mathcal{A}|$, the number of agents, and $p_{edge} \in (0, 1)$, the probability of an edge. We created a graph $G = (V_G, E_G)$ such that for each possible edge $e = \{v_1, v_2\}$, $e$ was added into $E_G$ if $p_{edge}$ was greater than a random number uniformly generated in $[0, 1]$. Then, we checked that all agents in the graph were connected, otherwise the graph was discarded and re-generated. In our experiments below, we iterated between values of $p_{edge}$ from $0.1, 0.2, \ldots, 0.9$ and collated the results across all $p_{edge}$.

Then, the simulator generated the agents' capabilities. In our first set of experiments, $M$, the number of sub-tasks, was set to 1. We generated $N_a \sim \mathcal{N}(\mu_a, \sigma^2_a) \in N_S$ such that $\mu_a \in [-\gamma, \gamma]$ and $\sigma^2_a \in [0, \gamma]$, where $\gamma$ is a multiplying factor, which we describe below. To generate the capabilities in the second set of experiments where $M > 1$, the $M$ sub-tasks were first split among the agents, such that $\lceil \frac{|\mathcal{A}|}{M} \rceil$ agents were capable of performing each sub-task. Then, when generating the Normal distributions, if an agent $a$ was capable of performing sub-task $m \in [1, M]$, then $N_{a,m} \sim \mathcal{N}(\mu_{a,m}, \sigma^2_{a,m})$, and $N_{a,m} \in N_a \in N_S$ such that $\mu_{a,m} \in (\frac{\gamma}{2}, \frac{3\gamma}{2})$, and $\sigma^2_{a,m} \in (0, \gamma)$. Otherwise, the distribution $N_{a,m} \sim \mathcal{N}(0, \epsilon)$ for some small $\epsilon$.

We created 2 weight functions $w$, $w_{frac}(d) = \frac{1}{d}$ and $w_{decay} = \exp(-\frac{d \ln 2}{h})$, where $d$ is the distance between 2 agents in the graph, and $h$ is the half-life of the exponential decay function. The two weight functions were selected to demonstrate that the algorithms' performance is similar regardless of the weight function, and because both $w_{frac}$ and $w_{decay}$ were intuitive and easy to understand. For the experiments in this paper, we set $h = 3$, since $|\mathcal{A}|$ was set to be at most 10, so the weight between agents would have a similar range for both functions.

$\gamma$, the multiplying factor, affects how the performance of the agents are affected by the weight functions. For example, a weight of $\frac{1}{2}$ reduces a capability of 4 to 2 (a difference of 2 units), but reduces 40 to 20 (a much larger difference of 20), which could have effects on the learning algorithm. Thus, we varied $\gamma$ in our learning experiments to study the effect of the range of utilities on the performance of our algorithm.

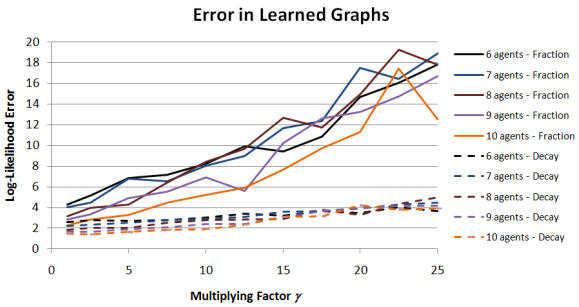


**Figure 5: The error in the learned graph with varying number of agents and both weight functions.**

### 5.1.1 *Learning Synergy Graphs from Observations*

In Sec. 4, we described the algorithm used to learn a synergy graph from observations. We first generated a synergy graph $S_{true}$ using the method described above, and then generated a training observation set $O_{train}$, with sets of 2 and 3 agents, i.e., $\forall\, O_A \in O_{train}$, $2 \leq |A| \leq 3$. We generated 100 data points from each pair/triple, and modeled a synergy graph $S_{learned}$ from the data.

Then, to test how well our algorithm learns the synergy graph, we generated a test observation set $O_{test}$ using combinations of 4 or greater agents, i.e., $\forall\, O_A \in O_{test}$, $|A| \geq 4$, that had 1000 observations in total. We then measured the difference in log-likelihoods between the hidden and learned synergy graphs, i.e., $LL(O_{test}|S_{true}) - LL(O_{test}|S_{learned})$, where $LL(O|S) =$ `CalculateLogLikelihood`$(O, S)$. A low log-likelihood difference indicates that the synergy graph is as likely as the true graph to have produced the observations. We compared this difference in log-likelihood versus the initial graph used in the learning algorithm, $S_{initial}$, that had random edges but learned agent capabilities, to observe if the graph structure has an effect on the log-likelihoods.

We first ran experiments where there was a single task ($M = 1$) and agents had heterogeneous levels of performance with regards to the task. Fig. 5 shows the log-likelihood differences of the learned synergy graphs with the 2 weight functions compared to the hidden synergy graph, and varying the number of agents from 6 to 10. Varying the number of agents does not affect the log-likelihood error much — the weight function and the multiplier $\gamma$ have greater effects.

Figs. 6 and 7 shows the log-likelihood difference of the learned synergy graphs and the initial synergy graphs when $|\mathcal{A}| = 10$. The learned synergy graphs are much closer to the true synergy graphs, i.e., the difference in log-likelihood is close to 0 and orders of magnitude lower than the initial synergy graphs. The error in log-likelihood increases as $\gamma$, the multiplying factor, increases, especially for $S_{initial}$, and shows that $\gamma$ affects the difficulty of the learning problem (seen from the errors of $S_{initial}$), but our learning algorithm is capable of significantly reducing this error in $S_{learned}$. Furthermore, the observation set used for learning only included pairs and triples of agents, but the learned graph had a low log-likelihood difference when testing against data of teams comprising 4 or more agents, which shows that the structure of the learned graph and the individual agent capabilities match the hidden synergy graph well.

The next set of experiments were run where the task was composed of $M > 1$ sub-tasks, and each sub-task had a number of agents that were capable of performing it. We

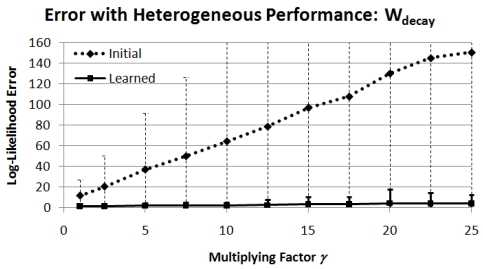


**Figure 6: The error in the learned synergy graph of 10 agents with heterogeneous task performance, using the weight function $w_{decay}(d) = \exp(-\frac{d \ln 2}{3})$, compared with the initial graph used by the learning algorithm, with random structure but learned agent capabilities.**

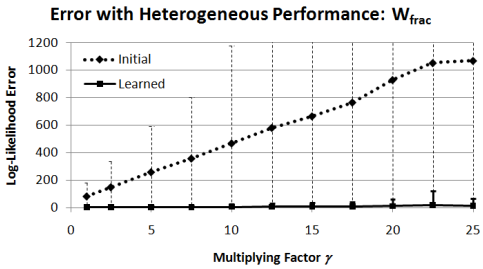


**Figure 7: The errors in synergy graphs of 10 agents with heterogeneous task performance, using the weight function $w_{frac}(d) = \frac{1}{d}$.**

used $w_{decay}$ as the weight function, set $|\mathcal{A}| = 10$, and varied $M$ from 2 to 5. Fig. 8 shows the log-likelihood differences between the $S_{learned}$ and the $S_{true}$, as compared to $S_{initial}$ and $S_{true}$. The error in the learned graphs are half or less than that of the initial graphs, which demonstrates the efficacy of our learning algorithm, using only data of 2 and 3 agents and being tested on larger groups of agents. However, while the error in log-likelihood of $S_{learned}$ remains mostly flat compared to $\gamma$, the error increases as $M$ increases, which shows that an increase in the number of sub-tasks has a large impact in the quality of the learned synergy graph. The advantage of $w_{decay}$ over $w_{frac}$ should be general, because the decay function decreases less abruptly as distance increases.

### 5.1.2 *Measuring Team Performance*

The goal is to find the optimal team to perform the task, and we use `ApproxOptimalTeam` (Algo. 1) on the learned synergy graphs $S_{learned}$. The performance of this set of agents is then computed with $F$ in the hidden synergy graph $S_{true}$, and compared against the best and worst possible combinations of agents. For example, if the set of agents $A'$ is selected from $S_{learned}$, then the value of $A'$ is computed on $S_{true}$.

We did two sets of experiments: when $M = 1$ and when $M > 1$. In the first set, we varied $|\mathcal{A}|$, the number of agents in the synergy graph, from 6 to 10, and the algorithm picked the best 5 agents. In the second set, we fixed $|\mathcal{A}|$ to 10 and varied $M$, the number of sub-tasks, from 2 to 5. In both cases, $\gamma = 1$, and $\rho = 0.75$. Tables. 1 and 2 show the score of the picked agents on a scale of 0 to 100 (where 0 denotes the worst possible combination, and 100 is the optimal team), and the performance of the selected team on the hidden


**Figure 8: The error in the initial and learned synergy graphs of 6 agents where some sub-tasks could only be completed by some agents, using the weight function $w_{decay}$.**

| | Learned Graph | Hidden Graph |
|---|---|---|
| 6 agents | $99.90 \pm 0.73$ | $100.00 \pm 0$ |
| 7 agents | $99.89 \pm 0.45$ | $100.00 \pm 0.05$ |
| 8 agents | $99.94 \pm 0.32$ | $100.00 \pm 0$ |
| 9 agents | $99.96 \pm 0.28$ | $100.00 \pm 0$ |
| 10 agents | $99.95 \pm 0.36$ | $99.99 \pm 0.12$ |

**Table 1: Score (%) of agents with 1 sub-task.**

| | Learned Graph | Hidden Graph |
|---|---|---|
| 2 sub-tasks | $99.64 \pm 0.77$ | $99.96 \pm 0.40$ |
| 3 sub-tasks | $99.57 \pm 3.43$ | $99.97 \pm 0.47$ |
| 4 sub-tasks | $98.79 \pm 9.95$ | $99.97 \pm 0.38$ |
| 5 sub-tasks | $89.65 \pm 30.26$ | $99.97 \pm 0.25$ |

**Table 2: Score (%) of agents: 10 agents with varying number of sub-tasks.**

graph $S_{true}$. The worst and optimal teams were discovered by iterating through all possible combinations of agents and computing their value. It is remarkable that our algorithm finds a team that obtains a score of at least 89.65%, and has a similar score when the algorithm is run on the hidden graph, and thus shows that the learned synergy graphs are in fact very close to hidden synergy graphs that were used to generate the observation sets, and that `ApproxOptimalTeam` can be used to find effective teams.

## 5.2 $F$ as a Probabilistic Function

In this section, $F$ was no longer a hidden synergy graph model. Instead, we used the robot capability model of Parker and Tang [8], where every robot has a subset of actions that it can perform, each with a probability of success. These actions are then chained across robots to produce the desired output, again with some probability of success. Fig. 9 shows the capabilities of 3 agents and how the actions are chained together to produce the desired outcome. Since each agent has a subset of the actions, different subsets of agents will have different results. In our experiments, we varied the number of agents from 4 to 10 and randomly picked their capabilities in each trial — each agent had a 0.7 chance of being able to perform each action, and the probability of success of the action was uniformly sampled from $[0.1, 0.9]$.

$F$, the function of the performance of a team of agents, was calculated based on the cost of executing the actions and the reward achieved by generating the output. The cost of attempting actions 1, 2 and 3 were 30, 10 and 15 respectively, and the cost of attempting to generate the output from action 1 was 10 and 15 from the combination of action 2 and 3. When the output was achieved successfully, a reward of 100 was given. The values of costs and reward were arbitrarily chosen, but further experiments with different values yielded similar results, so we present these results in this paper. In each trial, every agent would attempt to execute its actions, and if they were successful, the output was also attempted to be generated. Thus, $F$ had a probability density function (pdf) that depended on the agent capabilities — this pdf was not Normally distributed in general. We were interested to find out how accurately we could learn a synergy graph to model $F$ even in such a situation.

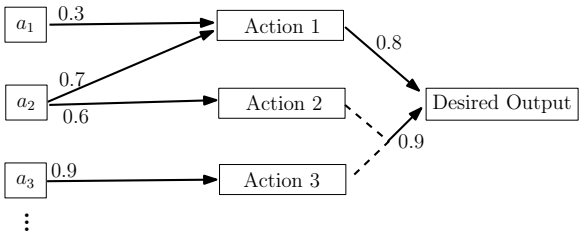


**Figure 9: The model of robot capabilities introduced by Parker and Tang [8]. The numbers indicate probabilities of success, and the dashed lines out of actions 2 and 3 indicate that both are required to trigger the desired output.**

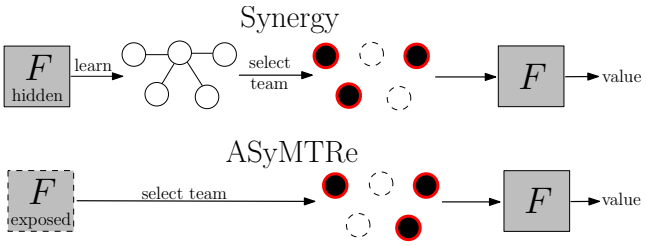


**Figure 10: The experimental process to compare our synergy algorithm against the ASyMTRe algorithm.**

Fig. 10 shows the experimental process. The hidden function $F$ was used to generate observations of subsets of 2 and 3 agents, and then a synergy graph model is learned (Algo. 2). A team is then selected using the learned synergy graph (Algo. 1), and the value of the team is computed using $F$. To attain results for ASyMTRe [8], the probabilities of success and costs of actions in $F$ were exposed, and the heuristic to rank teams in [8] was used. The ASyMTRe algorithm is an anytime algorithm, but for our experiments, we ran it to completion so that the optimal team with respect to the authors' heuristic was chosen. The values of the selected teams were then compared to the maximum and minimum team values, which were attained by performing a brute-force search of all possible combinations of agents in $F$, and thus scaling the results of the synergy and ASyMTRe algorithms to be between 0 and 1.

The ranking heuristic in the ASyMTRe algorithm has a factor $p \in [0, 1]$ that balances between the probability of success of performing an action versus the cost of the action. For our experiments, we varied $p$ from 0 to 1 at 0.1 intervals, and collated the results. Similarly, the synergy algorithm uses the risk factor $\rho \in (0, 1)$; we varied $\rho$ from 0.1

| # of agents | Synergy | ASyMTRe |
|---|---|---|
| 4 | $95 \pm 17$ | $64 \pm 34$ |
| 5 | $95 \pm 14$ | $64 \pm 33$ |
| 6 | $96 \pm 10$ | $63 \pm 31$ |
| 7 | $97 \pm 8$ | $60 \pm 29$ |
| 8 | $93 \pm 7$ | $59 \pm 27$ |
| 9 | $97 \pm 7$ | $59 \pm 25$ |
| 10 | $96 \pm 8$ | $63 \pm 27$ |

**Table 3: Score (%) of teams composed by our synergy algorithm versus the ASyMTRe algorithm.**

to 0.9 at 0.1 intervals and collated the results. The results were collated across $p$ and $\rho$ since the values were consistent and had little effect on the performance of the algorithms in general. We varied $|\mathcal{A}|$, the number of agents, from 4 to 10, and picked teams of sizes 2 to $|\mathcal{A}| - 1$. For a given size of $|\mathcal{A}|$, we performed 30 trials for each team size.

Table 3 shows the scores of the two algorithms. Across all number of agents, our synergy algorithm outperforms the ASyMTRe algorithm in terms of the performance of the team selected, even though the function $F$ is hidden to the synergy algorithm but exposed for the ASyMTRe algorithm. The ASyMTRe algorithm finds teams that score around 60% of the optimal while the synergy algorithm forms teams that score above 90%. This significant difference is due to a number of reasons: firstly, the ASyMTRe algorithm was designed to also plan the agents' actions, i.e., which actions each agent should perform in order to complete the task. Secondly, the ASyMTRe typically plans for a set number of outputs (e.g., find a team to produce 2 outputs), but in our experiments the heuristic was used to find a team that produces as much output as possible. We compared our synergy algorithm to ASyMTRe as it is a well-known algorithm for multi-robot team formation and coordination that exploits heterogeneity in the agents to maximize task performance.

# 6. CONCLUSIONS

We are interested in team formation, where heterogeneous agents of varying capabilities are put together to complete a task. The interactions between these agents are initially unknown, and the goal is to select a subset of these agents such that the task performance is maximized.

We formally defined a synergy graph, where an agent's capability is represented by a list of Normal distributions, and the task-based relationship between agents are modeled by the distance between them in a graph. We then formally defined how pairwise synergy can be computed using a synergy graph, and extended the definition of synergy to include groups of any number of agents. We then presented an algorithm to approximate the optimal team given a synergy graph. Next, we contributed an algorithm that learns a synergy graph from observations of the performance of groups of agents, without any prior information about the agents' capabilities or the interactions among them. While we used simulated annealing in our team formation and learning algorithms, we believe that other approximation techniques would have similar performance.

In our experiments, we used 2 weight functions to weight the synergy of agents based on their distance in the graph. Using only observations of pairs and triples of agents, we showed that our learning algorithm is capable of learning the structure of task-based interactions and the capabilities of the agents as compared to the initial synergy graph where the observations were generated from, using both weight functions. This is a significant contribution as it shows that our learning algorithm does not need to observe all combinations of agent interactions in order to learn the synergy model, and is robust to different weight functions. Furthermore, we used a probabilistic model of agent capabilities to determine task performance, and compared our synergy algorithm with the ASyMTRe algorithm, and showed that even though the hidden model was not Normally distributed, and our algorithm does not have *a priori* knowledge of the agents' capabilities and probabilities of success of their actions while ASyMTRe has full knowledge, we are able to form teams that perform much better.

## Acknowledgments

This work was partially supported by the Air Force Research Laboratory under grant no. FA87501020165, 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.

# 7. REFERENCES

[1] B. Banerjee and L. Kraemer. Coalition structure generation in multi-agent systems with mixed externalities. In *Proc. 10th Int. Conf. Autonomous Agents and Multiagent Systems*, pages 175–182, 2010.

[2] C. Dorn and S. Dustdar. Composing near-optimal expert teams: A trade-off between skills and connectivity. In *Proc. Int. Conf. Cooperative Information Systems*, pages 472–489, 2010.

[3] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *Int. J. Robotics Research*, 23(9):939–954, 2004.

[4] C. Guttmann. Making allocations collectively: Iterative group decision making under uncertainty. In *Proc. 6th German Conf. Multiagent System Technologies*, pages 73–85, 2008.

[5] L. He and T. R. Ioerger. A quantitative model of capabilities in multi-agent systems. In *Proc. Int. Conf. Artificial Intelligence*, pages 730–736, 2003.

[6] T. Lappas, K. Liu, and E. Terzi. Finding a Team of Experts in Social Networks. In *Proc. Int. Conf. Knowledge Discovery and Data Mining*, pages 467–476, 2009.

[7] S. Liemhetcharat and M. Veloso. Mutual state capability-based role assignment model (extended abstract). In *Proc. 9th Int. Conf. Autonomous Agents and Multiagent Systems*, pages 1509–1510, 2010.

[8] L. Parker and F. Tang. Building multirobot coalitions through automated task solution synthesis. *Proc. IEEE*, 94(7):1289–1305, 2006.

[9] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111:209–238, 1999.

[10] T. Service and J. Adams. Coalition formation for task allocation: theory and algorithms. *Autonomous Agents and Multi-Agent Systems*, 22:225–248, 2011.