Modeling Mutual Capabilities in Heterogeneous Teams for Role Assignment

Somchaya Liemhetcharat and Manuela Veloso
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
som@ri.cmu.edu and veloso@cs.cmu.edu

***Abstract*— The performance of a heterogeneous team depends critically on the composition of its members, and switching out one member for another can make a drastic difference. The capabilities of an agent depends not only on its individual characteristics, but also the interactions with its teammates. Roles are typically assigned to individual agents in such a team, where each role is responsible for a certain aspect of the joint team goal. In this paper, we focus on role assignment in a heterogeneous team, where an agent's capability depends on its teammate and their mutual state, i.e., the agent's state and its teammate's state. The capabilities of an agent are represented by a mean and variance, to capture the uncertainty in the agent's actions and in the world. We present a formal framework for representing this problem, and illustrate our framework using a robot soccer example. We formally describe how to compute the value of a role assignment policy, as well as the computation of the optimal role assignment policy, using a notion of risk. Further, we show that finding the optimal role assignment can be difficult, and describe approximation algorithms that can be used to solve this problem. We provide an analysis of these algorithms in our model and empirically show that they perform well in general problems of this domain, compared to market-based techniques. Lastly, we describe an extension to our proposed model that captures mutual interactions between more than two agents.**

I. Introduction

Role assignment in heterogeneous teams, i.e., teams with members of different capabilities, has been an extensive topic of research. Different capabilities among members make some of them better suited for specific roles, and role assignment algorithms plan based on these capabilities. Common approaches to this problem, which we further discuss in the related work section, include market-based techniques, where agents bid over the roles, and the capabilities and states of the agents are used in generating the bids. Most current approaches that explicitly model capabilities assume that the agents have fixed capabilities for tasks, which may not be an accurate reflection in dynamic environments, since the state of the world affects an agent's ability to perform an action.

We are interested in explicitly modeling the capabilities of the agents in dynamic environments. We address the case where an agent's ability in performing an action depends on the teammate that is affected by the action, as well as their mutual state, i.e., the agent's state and its teammate's state. Modeling capabilities in this way captures information about both the innate abilities of the agent in performing the action, as well as how effective a particular pairing of agents would be when the action is taken in their mutual states. For example, an agent may perform an action well with a particular teammate in some mutual states, but not in other mutual states, or with any other agent. Alternatively, any two agents may work well together at one particular mutual state. We contribute a model that captures these situations.

We are also interested in a ad-hoc scenario, where agents who have had no prior contact are put together to form a team. The agents do not have prior knowledge of the capabilities of their teammates, nor how well they work together. Thus, we assume that information about capabilities is incrementally obtained through interaction and observation, which implies that agents are aware of their actions and learn about the actions' effects. In this paper, we formally model the capabilities of the agents, and do not discuss the techniques used to learn the capabilities from observations.

Since the capabilities of agents are incrementally obtained, we define the capability of an agent performing an action (with a particular teammate in a mutual state) as the mean and variance of the utility achieved by the action. Representing a capability with a mean and variance captures both the uncertainty in the agent's action and the uncertainty in the world. In addition, we introduce the concept of risk in role assignments, by evaluating the mean and variance of role assignment policies. Having a parameter for risk allows a tradeoff between the mean and variance in the optimal role assignment policy, so conservative role assignment policies are chosen over riskier ones, i.e., policies with higher mean utilities but correspondingly higher variances, if so desired. For example, by varying the amount of risk over time, an ad-hoc team of robots can quickly learn to work together and discover the best role-assignment of the entire team.

We contribute the Mutual State Capability-Based Role Assignment (MuSCRA) model, which includes the states of the world, the actions that can be taken, the roles in the team, and the capabilities of the agents. Roles have association strengths with states of the world, i.e., the states that are important to the role, and roles have emphases on actions, i.e., which actions are important to the role. We introduce the concept of an optimal role assignment in our model, and discuss approximation algorithms that can be used to find effective role assignments. We compare the performance of the approximation algorithms with a market-based approach.

There are several real scenarios in dynamic environments where MuSCRA can be applied. In robot soccer, robots may have different capabilities in kicking the ball accurately and

passing the ball to one another. Role assignment in this case would involve choosing the "best" players for the team (if there were more robots than actual players allowed on the field), and giving them the right assignments such as attacker or defender. The risk parameter would determine whether an aggressive role assignment is preferred over a conservative one. Similarly, in urban search and rescue, different robots have different capabilities, for example the ability to cross rough terrain, the speed of movement, and the ability to detect trapped humans. Furthermore, such abilities could depend on the composition and state of a team, e.g., a robot's ability to detect a human accurately could depend on the sensor readings of another robot. Roles could be diggers that clear the path for other robots, and searchers that travel along the cleared path to search for humans.

While pairwise interactions between agents are modeled in MuSCRA, it does not model interactions between larger groups of agents. We also contribute a model of mutual capabilities between larger groups of agents, which we term *synergy* to distinguish from the pairwise mutual capabilities in MuSCRA. The synergy model uses a graph to model the task-based interactions between the agents in the team, and the performance of a team depends on the agents' individual capabilities and the structure of the graph.

II. Related Work

Task allocation in multi-agent systems is very similar to role assignment, where the tasks in the domain can be likened to the actions in our approach, and roles represent responsibilities over certains groups of tasks. Gerkey and Mataric provide a detailed taxonomy of task allocation in multi-robot systems [1]. The authors also use the concept of utility, which they posit "is carried out somewhere in every autonomous task allocation system". Our approach uses utility, but instead of a single value, we use a mean and variance to represent the utility's distribution instead.

Market-based approaches are frequently used in task allocation domains, and Dias et al. provide a comprehensive survey [2]. Also, market-based approaches allow agents to perform role assignment [3]. While market-based approaches typically can determine a role assignment in $O(n)$ time or better (where n is the number of agents), the complexity of the domain is delegated to the method of generating bids. Instead of encapsulating this complexity, our approach explicitly models capabilities to perform role assignment.

Guttmann models uncertainty in the capabilities of teams in performing tasks, using means and standard deviations [4]. We model capabilities probabilistically as a function of the agent, a teammate, and their mutual state. Kok and Vlassis use states to model teammates to coordinate actions [5]; Gmytrasiewicz and Doshi model agents to select optimal actions [6]; Garland and Alterman study conventions to coordinate actions [7]. We model mutual capabilities to select a role assignment to form a team with a high utility.

III. Modeling Agent Capabilities

A heterogeneous team of agents consists of agents with different capabilities, and the goal is to find an assignment of roles for the agents such that the best team configuration is achieved, in terms of the utility attained. The capability of an agent to successfully perform an action with a teammate depends on their mutual state, i.e., the agent's state as well as the teammate's. These capabilities are assumed to be obtained incrementally from observation, and are represented by the mean and variance of the utility of performing the action. Role assignment of a team of robots incorporates a risk factor, which represents the tradeoff between the mean and variance of the utility of a role assignment policy.

A. The MuSCRA Model

Definition 1: A **Mutual State Capability-Based Role Assignment** (MuSCRA) is a tuple $\{\mathcal{X}, \mathcal{A}, \mathbf{a}, \mathcal{R}, \mathrm{S}, \mathrm{E}, \mathrm{C}, \rho\}$

- $\mathcal{X}$ is the set of states
- $\mathcal{A}$ is the set of actions
- $\mathbf{a}$ is the set of agents
- $\mathcal{R}$ is the set of roles
- $\mathrm{S} : \mathcal{R} \times \mathcal{X} \rightarrow [0, 1]$ is the association of roles and states
- $\mathrm{E} : \mathcal{R} \times \mathcal{A} \rightarrow [0, 1]$ is the emphasis of actions in roles
- $\mathrm{C} : \mathbf{a} \times \mathcal{X} \times \mathcal{A} \times \mathbf{a} \times \mathcal{X} \rightarrow (\mu_{\mathrm{C}}, \sigma^2_{\mathrm{C}})$ is the function of capabilities, where
$$\mathrm{C}(a_1, x_1, A, a_2, x_2)$$
returns the mean and variance of the utility obtained when agent a_1 in state x_1 performs action A while agent a_2 is in state x_2
- $\rho \in (0, 1)$ is the risk to take in assigning roles

To aid in the explanation of the MuSCRA model, we use a single scenario extensively. Consider a robot soccer scenario where a team of 2 robots plays against another team of 2 robots. The goal of each team is to score as many points into the opponent's goal as possible, while minimizing the number of points scored against themselves. Although the examples involve 2 agents in a robot soccer scenario, the MuSCRA model is applicable to scenarios with many agents, and in other domains, which we discuss later.

States: The set of states $\mathcal{X}$ is the set of all possible states of the agents. $\mathcal{X}$ is not the joint state-space of the team — each $x \in \mathcal{X}$ represents a state that a single agent can be in.

In robot soccer, the state of each robot has 2 features: its physical position on the field (defensive/offensive half), and whether it is blocked by an opponent. However, for ease of explanation, we only use the first feature; we will elaborate on the 4-states case (using both features) later in this section. We thus define the set of states: $\mathcal{X} = \{x_d, x_o\}$, where the x_d means that the robot is in the defensive half, and x_o means that the robot is in offensive half.

Actions: The set of actions $\mathcal{A}$ represents actions that the agents can take. Similar to $\mathcal{X}$, $\mathcal{A}$ is not the joint action-space of the team, but instead individual actions that each agent can take. In the robot soccer scenario, each robot can perform 3 actions, with varying capabilities. These actions (and their corresponding symbols) are: dribble (A_d), pass (A_p), and score (A_s). Thus, the set of actions is: $\mathcal{A} = \{A_d, A_p, A_s\}$.

Dribbling involves the robot moving the ball as along the field (typically around an opponent), while passing involves kicking a ball to a teammate. Scoring involves shooting

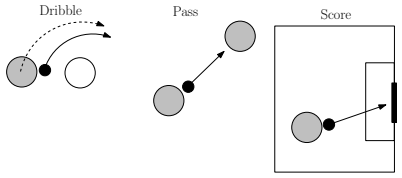

Fig. 1. Actions of soccer robots. Large circles indicate the robots, and small black circles indicate the ball. The thick bold line represents the goal area. Solid and dotted arrows indicate the path of the ball and robot respectively.

the ball directly at the opponent's goal. Fig. 1 shows an example of these actions in the field. The solid arrows indicate the direction of travel of the ball, while the dotted arrows represent the path of the robot.

In most situations, the utility obtained by an agent performing an action is affected by its current state, as well as its teammate and its teammate's state. For example, it is more likely for a robot in an open position to successfully pass to a teammate who is also in an open position, while it is more unlikely to succeed if either or both of the robots are blocked. Also, a particular robot may be better at receiving passes than other robots in the team.

Agents: The set of agents, $\mathbf{a}$, represent the team of cooperative agents whose roles are being assigned. Only the team of agents are considered — adversarial agents (or agents that cannot be controlled) are not part of this set.

Roles: The roles of the team, $\mathcal{R}$, represent associations with certain states of the world, as well as an emphasis in certain actions. Each role is assigned to a single agent, and so roles can be viewed as the smallest element of a team. In particular, the number of roles should be no larger than the number of agents, i.e., $|\mathcal{R}| \leq |\mathbf{a}|$, which ensures that every role can be fulfilled by an agent. In robot soccer, we can define the set of roles as: $\mathcal{R} = \{R_d, R_a\}$, where R_d is a defender and R_a is an attacker. The defender's responsibility is primarily to prevent a goal from being scored against the team, and secondarily to pass the ball upfield to the attacker. The attacker's responsibility is to score goals for the team.

Association between Roles and States: Roles are associated with states of the world, and this is represented by the function $\mathrm{S} : \mathcal{R} \times \mathcal{X} \rightarrow [0, 1]$, which indicates how strongly associated a state and role are, where a higher value indicates a stronger association. S is such that $\forall\ R \in \mathcal{R},\ \sum_{x \in \mathcal{X}} \mathrm{S}(R,\ x) = 1$. Thus, the sum of all associations of any given role is 1, which provides normalization so that the weighting of every role is equal. A state x that is unassociated with any role, i.e., $\forall\ R \in \mathcal{R},\ \mathrm{S}(R, x) = 0$, means that the state is unimportant to all roles in the team.

Emphasis of Actions in Roles: Different roles may emphasize different actions, to indiciate how important the action is to a role. This emphasis is given by the function $\mathrm{E} : \mathcal{R} \times \mathcal{A} \rightarrow [0, 1]$, such that $\forall R \in \mathcal{R}, \sum_{A \in \mathcal{A}} \mathrm{E}(R, A) = 1$, which states that the sum of emphases for any role is 1, which ensures normalization.

In our robot soccer example, the defender role R_d places a high emphasis on the dribbling action A_d and the passing action A_p, with a little emphasis on the scoring action A_s. This reflects the defender's role in preventing a goal from being scored and pushing the ball upfield. The small emphasis on scoring reflects that the defender may take a direct shot on goal if an opportunity arises. The attacker role R_a places a very high emphasis on scoring (which reflects its main responsibility), and dribbling (which it uses to go around opponents). There is no emphasis on passing, since there is no other player upfield worth passing the ball to.

Mutual State Capabilities: In a heterogeneous team, agents have different capabilities in the actions and skills relevant to the task. The capability function, $\mathrm{C} : \mathbf{a} \times \mathcal{X} \times \mathcal{A} \times \mathbf{a} \times \mathcal{X} \rightarrow (\mu_{\mathrm{C}},\ \sigma^2_{\mathrm{C}})$, represents these capabilities of the agents.

The first two parameters of C represent the agent and its state. The third parameter, $\mathcal{A}$, represents the action being performed, and the last two parameters represent the teammate and the teammate's state. Such a parametrization of capabilities takes into account that an agent's ability to perform an action and achieve the desired outcome depends on the teammate, and their mutual state, i.e., the agent's state and the teammate's state. If an action does not depend on a teammate, e.g., the scoring action A_s in the robot soccer example, then the values returned by C is equal for all values of teammates and teammate states.

C returns a mean and variance, which represent the distribution of the utility obtained by performing the action. The variance in the utility is affected by the uncertainty in performing the action (how likely the robot is able to carry out the action) as well as the uncertainty in the world (how likely the desired outcome of the action is obtained).

In addition, the means and variances returned by C are intended to be obtained through a data-driven approach, e.g., learning or modeling. Thus, by including a variance term, the model captures an idea of how much data has been collected and how consistent a robot is in its actions. For example, if only the mean was captured, then the value of a robot that is consistently observed to be average would be equal to that of another robot that has only been observed twice, once really well and once really poorly.

We henceforth adopt the notation C_μ to represent the function derived from C that returns only the mean (μ_{C}), and C_{σ^2} to represent the function derived from C that returns only the variance (σ^2_{C}).

Risk: The ρ term in the MuSCRA model represents how much risk to take while assigning roles to the team. The utility U_r of a role assignment is normally distributed with a mean and variance, and given a certain value of $\rho \in (0, 1)$, u is a value such that $U_r \leq u$, i.e., $P(U_r \leq u) = \rho$. Thus, as ρ increases, the probability that the role assignment value takes on a lower value than u increases, and so ρ is a measure of risk taken in the role assignment. Further details on using ρ in calculate the value of a policy is shown later.

B. Applications of the MuSCRA Model

The robot soccer scenario described in detail is a good application of the MuSCRA model. The states of each robot can be extended to have two main features, its position on the field, and whether the robot is blocked by an opponent.

Such a formulation encapsulates more information about the world. For example, the capability of a robot passing a ball to a teammate would take into account the robots' abilities to manoeuvre around the opponents, and the opponent's abilities to get in the way.

Table I lists a number of domains that the MuSCRA model can be applied to. Besides robot soccer, the MuSCRA model works well in other domains that involve role assignment to heterogeneous teams, e.g., urban search and rescue (USAR). The success of agents' actions in USAR depends on their teammates and their mutual states, e.g., the ability for a robot to move through a blocked area depends on which teammate is carrying rubble to help clear the path. Roles can be split into searchers, whose main job is to search for humans, and diggers, who clear rubble from the area and allow the searchers to travel quickly.

The MuSCRA model can also be applied to task allocation domains, such as in preparing a presentation. Actions include analyzing data for the presentation, creating slides, and actually making the presentation. These actions depend on teammates and mutual states as well. For example, an agent's capability in drawing graphics depends on whether it has data available and which teammate planned the slides' layouts.

Domain	State Features	Actions	Roles
Robot Soccer	Position on field Clear / Blocked by opponent	Dribble Pass Score	Defender Attacker
Urban Search & Rescue	Near human Blocked / Open area Carrying rubble	Detect humans Dig through rubble Carry rubble Move	Searcher Digger
Assembly	Holding object Near object Object in sight	Lift object Rotate object Drill Bolt	Coarse Manipulator Fine Manipulator
Presentation	Data available Graphics created Layout planned	Analyze data Draw graphics Create slides Make presentation	Slide creator Graphic Designer Presenter

TABLE I

EXAMPLES OF THE MUSCRA MODEL IN DIFFERENT DOMAINS

IV. ROLE-ASSIGNMENT IN MUSCRA

In order to find the optimal assignment of roles in the MuSCRA model, we define a role assignment policy:

Definition 2: A **role assignment policy** $\pi : \mathcal{R} \rightarrow \mathbf{a}$ is an assignment of roles to agents such that every agent has at most 1 role, i.e., $\pi(R) = \pi(R') \Rightarrow R = R'$.

Def. 2 states that role assignments are unique; no agent has more than 1 role. When there are more agents than there are roles, it is possible for some agents to have no role. Role assignment policies are similar to the single-task robots, single-robot tasks, instantaneous assignment (ST-SR-IA) in Gerkey and Mataric's task allocation taxonomy [1].

A. Finding the Optimal Policy

Given a role assignment policy π, we determine the utility of the team thus assigned, taking into account the capabilities of each agent and its assigned role. We define the utility of a role assignment policy:

Definition 3: The **utility** of a role assignment policy is determined via the function: $U : \pi \rightarrow (\mu_\pi, \sigma^2_\pi)$, where μ_π and σ^2_π represent the mean and variance of the policy's utility.

We denote U_μ as the function derived from U that returns the mean, and U_{σ^2} as the function derived from U that returns the variance. The functions are computed as follows:

$$U_\mu(\pi) = \sum_{\substack{R,R' \in \mathcal{R}: R \neq R' \\ x,y \in \mathcal{X} \\ A \in \mathcal{A}}} \phi(\cdot) C_\mu(\pi(R), x, A, \pi(R'), y)$$

$$U_{\sigma^2}(\pi) = \sum_{\substack{R,R' \in \mathcal{R}: R \neq R' \\ x,y \in \mathcal{X} \\ A \in \mathcal{A}}} \phi(\cdot) C_{\sigma^2}(\pi(R), x, A, \pi(R'), y)$$

where $\phi(R, x, A, R', y) = \mathrm{E}(R, A)\mathrm{S}(R, x)\mathrm{S}(R', y)$ is a weight function.

Using the action emphasis function E and role-state association function S, ϕ determines how much weight to place on the utility of an action taken by a role. Thus, actions with more emphasis in the role will reflect a higher weight in ϕ. Similarly, highly associated states of the agent and its teammate will have higher weights during U's calculation.

It may seem that the calculation of a policy's utility involves a massive amount of computation. However, the states in which the roles are valid (i.e., $\mathrm{S}(R, x) > 0$ or $\mathrm{S}(R', y) > 0$) are typically much smaller than the entire state space. Thus, the computation of U can be optimized. Furthermore, the calculation of a policy's mean utility and variance is only a constant factor increase as compared to calculating the policy's mean utility alone.

Incorporating Risk into Utility: The utility of a policy π is normally distributed with a mean and variance, as computed by U_μ and U_{σ^2} respectively. Given the risk parameter ρ, we define the value of a policy as follows:

Definition 4: The **value** of a policy is given by the function $V : \pi \rightarrow \mathbb{R}$, where $V(\pi) = U_\mu(\pi) + \sqrt{U_{\sigma^2}(\pi)}\Phi^{-1}(\rho)$ and Φ^{-1} is the inverse of the standard normal cumulative distribution function.

Thus, ρ is the probability that the value of a role assignment policy π is lower than $V(\pi)$, and so ρ represents the risk taken in the role assignment policy.

Optimal Role-Assignment Policy: With a value function V as defined above, we define the optimal (or best) policy:

Definition 5: The **optimal policy** π^* is the policy with the highest value, i.e., $\forall \pi, V(\pi^*) \geq V(\pi)$.

B. Approximation Algorithms

Finding the optimal policy π^* using a brute-force method involves searching the entire space of policies, which is factorially large. As such, we consider approximation algorithms such as hill-climbing.

Hill-Climbing: To perform hill-climbing or other approximation techniques, we define neighbors of policies:

Definition 6: Role assignment policies π_1 and π_2 are **neighbors** if:

- π_1 and π_2 have a swapped value, i.e., $\exists x_i, x_j$ s.t. $(\pi_1(x_i) = \pi_2(x_j)) \wedge (\pi_1(x_j) = \pi_2(x_i)) \wedge (x_i \neq x_j) \wedge (\forall x \neq x_i, x \neq x_j, \pi_1(x) = \pi_2(x))$
  OR
- π_1 and π_2 differ by one value, i.e., $\exists x$ s.t. $(\pi_1(x) \neq \pi_2(x)) \wedge (\forall x' \neq x, \pi_1(x') = \pi_2(x'))$

Hill-climbing is performed by selecting an initial policy π and evaluating its neighbors and choosing the best neighbor to iterate on. It continues until all neighbors of the policy have equal or lower value.

	Defensive	Mid-Field	Offensive
Defenders	0.8	0.2	0
Mid-Fielders	0.2	0.6	0.2
Attackers	0	0.2	0.8

TABLE II

ROLE-STATE ASSOCIATIONS FOR ROBOT SOCCERS

	Pass	Clear	Dribble	Score
Defenders	0.2	0.5	0.3	0
Mid-Fielders	0.4	0.2	0.3	0.1
Attackers	0.2	0	0.2	0.6

TABLE III

ESSENTIAL ACTIONS IN ROLES FOR ROBOT SOCCER

Hill-Climbing with Random Restarts: The performance of the hill-climbing algorithm is dependent on the initial starting policy. In order to circumvent this issue, we can perform hill-climbing with random restarts. The algorithm calls the hill-climbing function repeatedly, starting with random initial policies, and returns the best policy from all the hill climbs.

V. EXPERIMENTS AND RESULTS

In order to test our model and solving algorithms, we first generated a robot soccer scenario, with different types of agents, and used the approximate algorithms as a proof-of-concept of our model. Next, we generated random instances of the MuSCRA model, and evaluated the performance of the approximate algorithms versus a market-based bidding technique. While the capability function is intended to be created via a data-driven (or learning) method, the focus of this paper is on the MuSCRA model and not the learning technique, and as such, we generated the capabilities based on normal distributions, as described below for each domain.

A. Robot Soccer Domain

To simulate robot soccer, we defined 3 states in the world: a defensive zone, mid-field, and an offensive zone. We defined 4 possible actions: passing, clearing, dribbling and scoring. Next, we created 3 **types** of roles for the agents: defenders, mid-fielders, and attackers. Given the number of agents n as input, we created $\lfloor n/3 \rfloor$ copies of the mid-fielder role, $\lfloor \frac{n-\lfloor n/3 \rfloor}{2} \rfloor$ defender roles, and the rest as attacker roles. As such, there were n roles to be assigned to n agents.

Table II shows the role-state associations for the defender, mid-fielder, and attacker roles, and Table III shows the emphasis of actions for the roles. There were **multiple copies** of each role type (defender, mid-fielder, attacker), and they each took on the relevant values shown in Tables II and III.

We generated the capabilities of the n agents by doing the following: given the number of defender, mid-fielder and attacker roles, we first pre-assigned each agent to a role. Next, we generated capabilities such that agents pre-assigned as defenders had higher mean utilities for the clearing action, mid-fielders had higher mean utilities for passing, and attackers had higher mean utilities for scoring.

The goal of this setup was to confirm that the approximation algorithms would be able to find the "right" role assignments, given the capabilities that we generated. We ran the hill-climbing algorithms, varying the number of robots from 3 to 7. In all cases, we found the optimal role assignment policy, which matched the pre-assigned roles of the agents.

B. Random Domains

We created a simulator in Java that, given as input the number of states $|\mathcal{X}|$, the number of actions $|\mathcal{A}|$, the number of agents $|\mathbf{a}|$ and the number of roles $|R|$, generates the functions S, E, and C randomly, as described below.

To generate the associations S between roles and states, we did the following: for each role and state, we generated a random number uniformly distributed between 0 and 1. We then normalized the values for each role, so that the sum of its associations was 1, i.e., $\forall\, R \in \mathcal{R},\, \sum_{x \in \mathcal{X}} \mathrm{S}(R, x) = 1$.

We created the emphasis function E similarly, by generating a random number uniformly distributed between 0 and 1 for each role-action pair. We then normalized these values as well to ensure that $\forall R \in \mathcal{R}, \sum_{A \in \mathcal{A}} \mathrm{E}(R, A) = 1$.

Lastly, we created the capabilities of robots by generating a mean between -1 and 1 (using a standard Normal distribution, with tail ends removed), and a variance between 0 and 1, taking the absolute value of a standard Normal distribution, with tail ends removed.

C. Evaluating the Algorithms

To compare the approximate algorithms against a market-based approach, a bidding technique had to be devised. The agents would bid for each role sequentially, and the agent with the highest bid is assigned the role. To generate the bid, each agent a calculated its individual capability for a role R, as shown below:

$$c(a, R) = \sum_{\substack{A \in \mathcal{A} \\ x \in X}} \mathrm{E}(R, A)\mathrm{S}(R, x) \frac{\sum_{\substack{a' \in \mathcal{A} \\ x' \in \mathcal{X}}} C(a, x, A, a', x')}{\sum_{\substack{a' \in \mathcal{A} \\ x' \in \mathcal{X}}} 1}$$

Using the mean and variance of its individual capability, each agent would then form the bid by incorporating the risk factor ρ. Essentially, each robot calculated its capability based on the average of its teammates and their states, since at the time of bidding, the agents do not have any knowledge about what roles their teammates will take.

We then implemented the hill-climbing and hill-climbing with random restarts algorithms and compared them with the bidding technique. To create a random policy, we chose a policy at random from the entire space of policies.

In the experiments for the random domains, we varied n to be between 5 and 7, and set $|\mathcal{X}| = |\mathcal{A}| = |\mathbf{a}| = |\mathcal{R}| = n$. We generated 300 MuSCRA models and ran the algorithms. We fixed the risk factor ρ to be 0.5 for all the experiments. The number of random restarts was set to be 5% of the number of possible policies. We limited the value of n to 7 since a brute-force search of large policy spaces (to obtain the optimal policy) would take extremely long to compute when $n > 7$.

We defined the effectiveness of the algorithm, by comparing the policy found π against the optimal policy π^* and the worst policy π_{min}, i.e., the policy with the minimum value:

$$effectiveness = \frac{V(\pi) - V(\pi_{min})}{V(\pi^*) - V(\pi_{min})}$$

Table IV displays the value of the best policy found by the algorithms, as a percentage compared with the brute-force method (which evaluates all policies). Hill-climbing and random restarts performed very well, compared to the market-based technique, without exploring much of the policy space (2.0% and 8.9% respectively when $n = 7$).

Effectiveness of policy in random domains	n		
	5	6	7
Hill-climbing	98.1%	97.5%	97.2%
Random restarts	98.3%	100%	100%
Market-based	68.1%	69.6%	70.2%

TABLE IV
EFFECTIVENESS OF POLICIES FOUND

To ensure that the risk factor ρ did not have an effect on the algorithms, we repeated the experiments on hill-climbing described above, fixing the value of n to 5, and varying ρ from 0 to 1, and running 1000 trials each. We found that the performance of the algorithms were not affected by the value of ρ, because the algorithms are blind to the value function, and the definition of neighbors in policy space allowed an effective exploration of the space. However, for a given random domain, different values of ρ will result in different optimal policies found.

VI. MODELING BEYOND PAIRWISE CAPABILITIES

In the previous sections, we formalized the MuSCRA model that incorporates pairwise interactions of agents. In this section, we describe an extension that models mutual capabilities between more than 2 agents. To distinguish between the MuSCRA model, we use the term *synergy* to refer to mutual capabilities among larger groups of agents.

A. Formalizing Synergy

In order to formally define synergy, we first introduce a graph that captures the synergy of a heterogeneous team:

Definition 7: 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:

- $v \in V_S$ is a vertex in G_S and represents an agent,
- E_S are unweighted edges in G_S, and
- $N_v \sim \mathcal{N}(\mu_v, \sigma_v^2) \in N_S$ is a Normal distribution representing the individual capability of agent v.

Every agent is represented by a vertex v in the graph, and the agent's individual capability is captured by a Normal distribution N_v. This distribution represents the agent's task performance, i.e., agents that are "better" at the task have higher means, and agents that are more "consistent" have lower variances, similar to the capability function C in the MuSCRA model. We will later extend this notion to agents with heterogeneous abilities, where some agents are capable of performing certain sub-tasks that other agents are not.

An edge in the graph G_S indicates that the pair of agents works together effectively. We will illustrate the use of our model with a search-and-rescue scenario. Suppose that a disaster has occurred, and many search-and-rescue teams have arrived. Some teams comprise entirely of humans, while others comprise a mix of semi-autonomous robots and human operators. Due to time and safety restrictions, not all teams

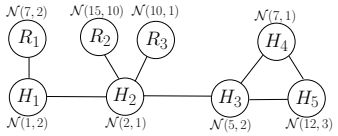

Fig. 2. A synergy graph of a search-and-rescue scenario. Vertices represent agents, and edges represent pairs of agents that work effectively together. The individual capabilities of agents are shown with Normal distributions.

are able to enter the disaster zone to perform search-and-rescue. Instead, only a small number of members can enter the area. Thus, the goal is to select the optimal subset of members from different teams, such that the search-and-rescue task is performed with the best task performance.

Fig. 2 shows a synergy graph of the search-and-rescue scenario. Each vertex represents an agent, i.e., the smallest team unit that can still perform the task. For example, R_1 is a 2-robot team that clears rubble, and H_5 is a 3-human team that specializes in freeing trapped victims. The individual capability of the agent is then the distribution of the task performance of that agent, e.g., how well R_1 clears rubble.

Edges in the graph indicate that agents work well together. For example, robot agents and their human operators have an edge (e.g., between R_1 and H_1 in Fig. 2), which represents the prior training and experience the pair of agents have had in the past. Similarly, edges between human agents (e.g., between H_3, H_4, H_5) indicate that they have trained together and are likely to coordinate well.

Thus, a pair of agents that are connected by an edge (e.g., H_4 and H_5 in Fig. 2) has high *synergy*. Conversely, if a pair of agents are "far" from each other in the graph, i.e., the distance between the pair of vertices is large, e.g., R_1 and H_5, then the pair will have low *synergy*. From this initutive description, we now formally define *pairwise synergy*:

Definition 8: The **pairwise synergy** $\mathbb{S}_2(v_i, v_j, S)$ between 2 agents v_i, v_j in S is a Normal distribution, given by $w(d(v_i, v_j, G_S)) \cdot (N_{v_i} + N_{v_j})$, where $d(v_i, v_j, G_S)$ is the shortest distance between the v_i, v_j in G_S, and $w(d)$ is a weight function that monotonically decreases as d increases.

The synergy between a pair of agents is defined above. However, in order to select the optimal team with the highest task performance, the synergy in a team with more than 2 members has to also be defined:

Definition 9: The **synergy** $\mathbb{S}(V_\alpha, S)$ of a set of agents $V_\alpha \subseteq V_S$ in S is the average of the pairwise synergy of its components, i.e., $\frac{1}{\binom{|V_\alpha|}{2}} \cdot \sum_{\{v_i, v_j\} \in V_\alpha} \mathbb{S}_2(v_i, v_j, S)$

B. Heterogeneity in Agent Abilities

In the previous subsection, each agent v had an individual capability N_v, that indicates its task performance. However, in many scenarios, the task may be so complex that no single agent is capable of performing the entire task. Instead, the task is split into sub-tasks which agents can complete. In heterogeneous teams, different agents have different capabilities, and are able to perform different sub-tasks. For example, in the search-and-rescue scenario, the task can be split into 3 sub-tasks, clearing rubble, finding victims, and rescuing victims. R_1, the 2-robot team that clears rubble, can perform the "clear-rubble" sub-task while H_5, the 3-human team that

specializes in freeing trapped victims, performs the "rescue" sub-task. In order to complete the task of search-and-rescue, all sub-tasks must be completed. To model the heterogeneous abilities, we extend the definition of the synergy graph:

Definition 10: An **extended synergy graph** $\tilde{S}$ is a tuple $\left\{G_{\tilde{S}}, \tilde{N}_{\tilde{S}}\right\}$, such that $G_{\tilde{S}} = (V_{\tilde{S}}, E_{\tilde{S}})$ is a connected graph, and $\tilde{N}_{\tilde{S}}$ is set of Normal distribution lists of length c, where:

- $\tilde{N}_v = (N_{v,1}, \ldots, N_{v,c}) \in \tilde{N}_{\tilde{S}}$ is a list of Normal distributions, i.e., the heterogeneous capability of v, and
- c is the number of heterogeneous capabilities, i.e., the number of sub-tasks.

Extended synergy graphs are similar to regular synergy graphs, except for the definition of $\tilde{N}_{\tilde{S}}$. Instead of a single distribution representing an agent's individual capability, each agent has a list of Normal distributions, which represent its capability in a particular sub-task.

C. Composing the Optimal Team

The goal is to find the optimal team $V^* \subseteq V_S$ from a synergy graph S. We assume that the size of the optimal team (i.e., $n^* = |V^*|$) is known. This is a reasonable assumption as the team size is often limited by other factors, such as monetary cost or environmental dangers in the search-and-rescue example. If n^* was unknown, then the team-forming algorithm can be run iteratively for increasing n, and the best performing one is picked as the final output.

To find the optimal team of n agents, a random team is first generated. Next, simulated annealing and/or hill-climbing 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, similar to the policy neighbor of the MuSCRA model. The score of the team is then computed using the synergy function $\mathbb{S}$, and iterated to obtain the optimal team.

VII. Conclusion

We formally defined the Mutual State Capability-Based Role Assignment (MuSCRA) model, and described each of its components in detail, using robot soccer as an example of an instantiation of the model. We briefly described other domains where MuSCRA can be applied, giving examples of possible states, actions and roles in those domains. Capabilities of agents in MuSCRA are defined not only as a pairing between an agent and action, but also incorporates the teammate, and their mutual state, i.e., the state of the agent and its teammate. This allows a generalization of capabilities to include the fact that the success of an action in a team depends on the composition of the team, as well as the state of the world. Capability is represented by a mean and variance, to signify the uncertainty in the actions and world.

We defined a role assignment policy, and how to determine the utility of such a policy, represented by a mean and variance. We then described how to incorporate the risk factor to retrieve the value of a policy. The risk factor adjusts the mean-to-variance trade-off in the optimal role assignment, and is an important contribution of our model. We then discussed approximation algorithms such as hill-climbing and hill-climbing with random restarts, ran extensive experiments on the approximation algorithms, and showed that hill-climbing (both the simple version and with random restarts) found effective policies (as compared to the optimal), performing better than market-based techniques. Different values of risk affected the optimal policy found, but did not affect the performance of the algorithms.

A possible application of MuSCRA is in an ad-hoc team scenario, where a group of cooperative robots collectively learn about their abilities in a team and discover an optimal role assignment for the team. However, there are certain drawbacks to our approach. Firstly, the calculation of a policy's utility can take extremely long in a worst-case scenario. Also, it may take a long time for the team to have sufficient observations to fill out the capability function before a good role assignment is obtained. An agent that is aware of its own skills may not be able to readily transfer this knowledge to the rest of the team, since the capabilities are defined as a function of the teammates.

We also contributed an extension to the MuSCRA model that captures the mutual capabilities of more than two agents, which we term synergy. The synergy model uses a graph structure to model the task-based interactions of the agents, and the performance of a team of agents depends on the identities and individual capabilities of the agents, as well as the structure of the graph that connects them.

Acknowledgments

This work was partially supported by the Lockheed Martin, Inc. under subcontract 8100001629/1041062, 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. This work is based on an earlier work: Mutual State-Based Capabilities for Role Assignment in Heterogeneous Teams, in Proceedings of the 3rd International Symposium on Practical Cognitive Agents and Robots, ACM, 2010. http://doi.acm.org/10.1145/1967112.1967115.

References

[1] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[2] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proc. IEEE*, vol. 94, no. 1, pp. 1257–1270, 2006.

[3] V. Frias-Martinez, E. Sklar, and S. Parsons, "Exploring auction mechanisms for role assignment in teams of autonomous robots," in *Proc. RoboCup Symp.*, 2004, pp. 532–539.

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

[5] J. Kok and N. Vlassis, "Mutual modeling of teammate behavior," Computer Science Institute, University of Amsterdam, The Netherlands, Tech. Rep. IAS-UVA-02-04, 2004.

[6] P. J. Gmytrasiewicz and P. Doshi, "A framework for sequential planning in multi-agent settings," *J. Artificial Intelligence Research*, vol. 24, pp. 24–49, 2004.

[7] A. Garland and R. Alterman, "Autonomous agents that learn to better coordinate," in *Proc. 3rd Int. Conf. Autonomous Agents and Multiagent Systems*, 2004, pp. 267–301.