2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Forming an Effective Multi-Robot Team Robust to Failures

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*— We are interested in forming a multi-robot team that attains high utility at a task, and is robust to failures in the robots. We consider configurable robots that are composed of modules, e.g., motors, sensors, and actuators, where each module has an independent probability of failure. The performance of the multi-robot team at the task depends not only on how the robots in the team are composed from modules, but also the probability of failure of the selected modules. We formally define the robust team formation problem, and introduce two methods of defining the optimal team. We contribute the Robust Synergy Graph for Configurable Robots (ρ-SGraCR) model, and two team formation algorithms to find effective robust teams. The first algorithm, *OptRobust*, runs in exponential time and finds the optimal robust team. The second algorithm, *ApproxRobust*, makes assumptions about the module failures and approximates the optimal robust team, and runs in polynomial time. We demonstrate the efficacy of the ρ-SGraCR model in modeling robust team performance, and evaluate *ApproxRobust* and *OptRobust*. Finally, we apply the ρ-SGraCR model to a real robot problem in the foraging domain, and show that it outperforms competing approaches.**

I. Introduction

Multi-robot teams provide redundancy so that the failure of a single robot does not completely stop the task from being performed. Previous work in robustness, that we detail in the related work section, has focused on coordination algorithms that are robust to failures. We are interested in forming robust multi-robot teams, using robots that are pre-defined with such coordination algorithms.

We consider configurable robots that are composed from modules such as motors, sensors and actuators. The failure of a module in a robot is modeled as a change in its composition, e.g., a robot with a wheeled base module and a radio module that experiences a failure in the radio module becomes a wheeled base module with no radio module. A multi-robot team is a set of such robots. We assume that module failures occur after the forming of the team (i.e., selecting the modules in each robot), but before the task is performed. Hence, the performance of a multi-robot team depends on the selected composition of modules in the robots, and the probability of module failures.

Our goal is to select the modules of every robot in the team, such that the performance of the team is robust even with failures in some modules. This general robust team formation problem is applicable to many robot domains, especially since robots are becoming increasingly modular. An example domain is foraging, where the goal is to search for and retrieve resources; modules include sensors that detect the resources and manipulators to forage them. Another example is missions in space, where the goal is to configure a multi-robot team to perform some mission in a remote location such as construction or exploration; modules include scientific instruments, manipulators and wheels.

In this paper, we differentiate between team *capability* and *performance*. Team capability is the utility attained by the team assuming all modules function successfully. Team performance is the distribution of utilities of the team, taking into account potential failures. We formally define the robust team formation problem, and contribute two methods of defining the optimal robust team: the risk-adverse optimal team and the risk-controlled optimal team.

To solve the robust team formation problem, we contribute the Robust Synergy Graph for Configurable Robots (ρ-SGraCR) model. ρ-SGraCR models the synergy (i.e., capability) between modules in a robot, as well as across robots. Using the module failure rate and synergy, the ρ-SGraCR model computes team performance. We contribute two robust team formation algorithms, *OptRobust* and *ApproxRobust*. *OptRobust* computes the optimal robust team of a given size in exponential time; *ApproxRobust* approximates the optimal robust team in polynomial time.

We evaluate the ρ-SGraCR model and robust team formation algorithms with extensive experiments. First, we show that the ρ-SGraCR model is capable of modeling domains where robustness increases with redundancy, as well as in domains where increasing the number of robots decreases robustness. Next, we compare *OptRobust* and *ApproxRobust*, and show that *ApproxRobust* finds effective robust multi-robot teams while running in polynomial time. We apply the ρ-SGraCR model to a real robot experiment in the foraging domain using Lego NXTs, CreBots (TurtleBots with CoBot software), and Aldebaran NAOs. We learn the ρ-SGraCR from experimental data, and use the learned model to form effective multi-robot teams that are robust to failure, and show that it outperforms competing approaches.

II. Related Work

One approach to modeling robot capabilities is to model robots and tasks as lists of services or resources [2], [3], and a multi-robot team is defined to be feasible to complete a task if the union of services/resources covers the task requirements. However, such approaches view multi-robot capabilities as whether a team can or cannot perform a task, e.g., [4]. We are interested in problems where the task performance varies as a function of the team composition.

Research in multi-robot robustness has mostly focused on coordination algorithms that are robust to failures, e.g., coordination strategies using self-organization and self-healing that complete a task even with robot failures [5]. Other ex-

amples include a distributed algorithm that increase failure-detection and robustness [6], and a multi-robot data association technique that improves performance in SLAM [7]. We are interested in forming a robust multi-robot team where the behaviors and coordination strategies of the robots are pre-defined — our approach forms the team that maximizes robustness given such algorithms.

Redundancy has been used to form robust multi-robot teams. Redundancy allows the team to stay in desired states when some robots fail [8], and ensures that at least one of the agents completes the task before a deadline [9]. We model and consider how the performance of the team varies with possible failures, in order to find a robust team.

III. Problem Definition and Approach

There is a complex task to be completed, and a robust multi-robot team is to be formed to perform the task. The goal is to form a *robust* team in the face of potential failures in the robots. To aid in the explanation of the problem, we will use a consistent motivating scenario. Suppose that the task is to forage resources from the environment.

The set of modules is $\mathcal{M} = M_1 \cup \ldots \cup M_N$, where each M_n is a set of modules of a certain type. A configurable robot R is a configuration of N modules, i.e., $R = (m_1, \ldots, m_N)$ where each $m_n \in M_n$. Hardware and software failures can occur, and we model the probability of success of modules:

Definition 3.1: The **probability of success** $0 \leq p_m \leq 1$ of a module $m \in \mathcal{M}$ is the probability that module m does not experience a failure.

Definition 3.2: The **fall-back module** m_n^* of type $n \in \{1, \ldots, N\}$ is a module that never fails, i.e., $p_{m_n^*} = 1$, and is the module that a robot $R = (m_1, \ldots, m_N)$ uses if it experiences a failure of module m_n.

For example, the robot $R = (m_1, \ldots, m_N)$ becomes $R = (m_1^*, m_2, \ldots, m_N)$ if module m_1 fails. In the foraging example, $N = 4$, where M_1 are motors, M_2 are sensors, M_3 are manipulators, and M_4 are robot bases. Let $R = (m_{\text{fast}}, m_{\text{camera}}, m^*_{\text{no manipulator}}, m_{\text{wheeled}})$ be an example of a wheeled robot capable of moving fast and detecting resources with its camera, but is unable to manipulate them. Fall-back modules can be defined to indicate the absence of something (e.g., $m_{\text{no base}}, m_{\text{no manipulator}}$) or provide a baseline module (e.g., m_{slow} for motors).

The probabilities p_m are independent, and multiple copies of the same module are also independent. For example, if $R_1 = (m, m')$ and $R_2 = (m, m'')$, a failure of m on R_1 is independent of a failure of m' in R_1, m and m'' in R_2.

The set of modules $\mathcal{M}$ and failure probabilities p_m are domain-specific and given as part of the problem description. The type of modules would typically be categorized by physical constraints (e.g., which modules can be physically placed simultaneously on a robot), and the success probabilities are determined by the mean time to failure of the hardware, or by experimentally determining the rates of failure.

$\mathcal{R}$ is the set of all robots, and $T = (R_1, R_2, \ldots) \in \mathcal{T}$ is a multi-robot team. A robot uses a fall-back module whenever a failure occurs in one of its modules.

Definition 3.3: The **set of alternative robots** $\mathcal{A}_{\mathcal{R}}(R)$ of a robot $R = (m_1, \ldots, m_N)$ is $\mathcal{A}_{\mathcal{R}}(R) = \bigtimes_{n=1}^{N} \{m_n, m_n^*\}$.

Definition 3.4: The **set of alternative teams** $\mathcal{A}_{\mathcal{T}}(T)$ of a team $T = (R_1, R_2, \ldots)$ is $\mathcal{A}_{\mathcal{T}}(T) = \bigtimes_{R \in T} \mathcal{A}_{\mathcal{R}}(R)$.

Definition 3.5: The **probability of occurrence of a robot** $R' \in \mathcal{A}_{\mathcal{R}}(R)$ is:

$$\mathbb{P}(R', R) = \prod_{m_n \in R} \begin{cases} p_{m_n} & \text{if } m_n \in R' \\ 1 - p_{m_n} & \text{otherwise} \end{cases}$$

Definition 3.6: The **probability of occurrence of a team** $T' = (R_1', R_2', \ldots) \in \mathcal{A}_{\mathcal{T}}(T)$ is:

$$\mathbb{P}(T', T) = \prod_{R_n' \in T'} \mathbb{P}(R_n', R)$$

Definition 3.7: The **capability** C_T of a team $T \in \mathcal{T}$ is the non-deterministic utility attained by T assuming that no failures occur in the modules of the robots comprising T.

Definition 3.8: The **performance** $\mathcal{P}_T$ of a team $T \in \mathcal{T}$ is the non-deterministic utility attained by T taking possible failures into account, and is a mixture model of the capabilities of the set of alternative teams of T: $\mathcal{P}_T$ has $|\mathcal{A}_{\mathcal{T}}(T)|$ components, where each component $T' \in \mathcal{A}_{\mathcal{T}}(T)$ has probability $\mathbb{P}(T', T)$ and distribution $C_{T'}$.

The robots act in a dynamic world where their actions have non-deterministic outcomes. C_T captures the performance due to the non-determinism in the world, e.g., wheel slippage causing a robot to arrive at a destination at a slower pace, and $\mathcal{P}_T$ captures both the non-determinism and the effects of failures. C_T and $\mathcal{P}_T$ are initially unknown, but some observations o_T of C_T are available.

Definition 3.9: The **robustness** $\rho(T, u)$ of a team $T \in \mathcal{T}$ is the probability that the performance $\mathcal{P}_T$ is at least a threshold utility threshold u: $\rho(T, u) = \mathbb{P}(\mathcal{P}_T \geq u)$.

Not all multi-robot teams are capable of completing the task, e.g., as a trivial example, a robot team where all the modules have failed cannot complete any task. $F : \mathcal{T} \rightarrow \{0, 1\}$ is the feasibility function, where $F(T) = 1$ iff the team T can complete the task. The feasibility function F is domain-specific and given.

Definition 3.10: The **risk-averse optimal team** T^*_{adv} is the team that has maximum robustness given a threshold threshold u_{thresh}: $T^*_{\text{adv}} = \text{argmax}_{T \in \mathcal{T} \text{ s.t. } F(T)=1} \rho(T, u_{\text{thresh}})$.

Definition 3.11: The **risk-controlled optimal team** T^*_{con} is the team with the highest utility with probability p_{con}: $T^*_{\text{con}} = \text{argmax}_{T \in \mathcal{T} \text{ s.t. } F(T)=1} \{u_T | \rho(T, u_T) = p_{\text{con}}\}$.

The risk-averse optimal team and risk-controlled optimal team are two sides of the same coin — the former maximizes robustness given a utility threshold, while the latter maximizes utility given a desired robustness. The problem domain determines whether the goal is to form the risk-averse optimal team or the risk-controlled optimal team (and the associated parameters u_{thresh} and p_{con} respectively).

We recently introduced the Synergy Graph for Configurable Robots (SGraCR) model [1], and we extend the SGraCR model to solve the robust team formation problem:

1) Use observations o_T of C_T to learn a SGraCR
2) Augment the learned SGraCR with p_m to form the ρ-SGraCR model
3) Form the optimal robust team using the ρ-SGraCR

IV. Multi-Robot Capability and Performance

The Synergy Graph for Configurable Robots (SGraCR) models the capability of configurable robots acting together in a multi-robot team [1]. We augment the SGraCR model to form the ρ-SGraCR model:

Definition 4.1: The **Robust Synergy Graph for Configurable Robots** model is a tuple $\{G, C\}$, where:

- $G = (V, E)$ is a connected weighted graph;
- Each vertex $v_m \in V$ represents a module $m \in \mathcal{M}$;
- Each vertex $v_m \in V$ is associated with a success probability $p_m \in [0, 1]$;
- $e = (v_m, v_{m'}, w_{\text{intra}}, w_{\text{inter}}) \in E$ is an edge with two integer weights: w_{intra} is the weight between modules on the same robot (intra-robot weight); w_{inter} is the weight between modules on different robots (inter-robot weight);
- $e_m = (v_m, v_m, w_{\text{inter}}) \in E$ is a self-looping edge with a single inter-robot weight;
- $C = \{C_1, \ldots, C_{|\mathcal{M}|}\}$ is a set of module capabilities, where $C_m \sim \mathcal{N}(\mu_m, \sigma_m^2)$ is the capability of $m \in \mathcal{M}$.

A. Computing Team Capability

Using the ρ-SGraCR model, we compute the capability of a multi-robot team, i.e., the utility attained by the team assuming that all modules are functional. We use the *intra-robot synergy* and *inter-robot synergy* functions [1]:

Definition 4.2: The **intra-robot synergy** $\mathbb{S}_{\text{intra}}(R)$ of a robot $R = (m_1, \ldots, m_N)$ is:

$$\mathbb{S}_{\text{intra}}(R) = \sum_{m_i, m_j \in R} \phi(d_{\text{intra}}(v_{m_i}, v_{m_j}))(C_{m_i} + C_{m_j})$$

where C_{m_i} and C_{m_j} are the module capabilities of m_i and m_j respectively, and $d_{\text{intra}}(v_{m_i}, v_{m_j})$ is the shortest intra-robot distance between vertices v_{m_i} and v_{m_j}.

Definition 4.3: The **inter-robot synergy** $\mathbb{S}_{\text{inter}}(R, R')$ of two robots $R = (m_1, \ldots, m_N)$ and $R' = (m'_1, \ldots, m'_N)$ is:

$$\mathbb{S}_{\text{inter}}(R, R') = \sum_{m_i \in R, m'_j \in R'} \phi(d_{\text{inter}}(v_{m_i}, v_{m'_j}))(C_{m_i} + C_{m'_j})$$

where C_{m_i} and $C_{m'_j}$ are the module capabilities of m_i and m'_j respectively, and $d_{\text{inter}}(v_{m_i}, v_{m'_j})$ is the shortest inter-robot distance between vertices v_{m_i} and $v_{m'_j}$.

$\phi : \mathbb{Z}^+ \rightarrow \mathbb{R}^+$ is a compatibility function that converts distances in the SGraCR graph to real valued compability. ϕ is a monotonically non-increasing function, so larger distances indicate lower compatibility.

To compute the capability of the entire multi-robot team, we use the synergy function [1]:

Definition 4.4: The **synergy** of a multi-robot team is:

$$\mathbb{S}(T) = \frac{1}{|T|} \sum_{R \in T} \mathbb{S}_{\text{intra}}(R) + \frac{1}{\binom{|T|}{2}} \sum_{R, R' \in T} \mathbb{S}_{\text{inter}}(R, R')$$

The synergy of the multi-robot team, i.e., its capability, is the average of the intra-robot synergy of the robots in the team, and the average of the intra-robot synergy between pairs of robots in the team. The synergy function uses the average, so as to prevent a bias towards larger teams having higher capabilities. Thus, $\mathbb{S}(T)$ aims to model $C(T)$, the initially unknown capability of team $T \in \mathcal{T}$.

V. Forming a Robust Multi-Robot Team

The SGraCR learning algorithm is general and does not require any domain-specific information, only observations o_T of C_T. Full details of the SGraCR model and learning algorithm are provided in [1]. From the learned SGraCR, we augment each vertex (that represents a robot module) with the probability of success p_m to form the Robust Synergy Graph for Configurable Robots (ρ-SGraCR) model. In this section, we contribute two team formation algorithms that use the ρ-SGraCR model to form a robust multi-robot team.

In our description below, we assume that the goal is to find the risk-adverse optimal team given a minimum threshold u_{thresh}, i.e., to find $T^*_{\text{adv}} = \text{argmax}_{T \in \mathcal{T} \text{ s.t. } F(T)=1} \rho(T, u_{\text{thresh}})$. The algorithms would only have to be modified slightly to find the risk-controlled optimal team.

Our first team formation algorithm, *OptRobust*, computes the optimal robust multi-robot team of size K. It is assumed that K is known, otherwise the algorithm is run iteratively for increasing K. Algorithm 1 shows the pseudo-code of *OptRobust*. The algorithm first generates all possible multi-robot teams comprising K robots. Next, the algorithm uses the synergy function $\mathbb{S}$ to compute the multi-robot team capability for all possible teams $\mathcal{T}$ and computes its score (based on the optimality function). To compute $\rho(T, u_{\text{thresh}})$, all combinations in $\mathcal{A}(T)$ have to be considered:

$$\rho(T, u_{\text{thresh}}) = \sum_{T' \in \mathcal{A}(T)} \mathbb{P}(T') \mathbb{P}(\mathbb{S}(T') \geq u_{\text{thresh}})$$

The number of teams of K robots is $|\mathcal{T}_K| = \mathcal{O}(|M|^{KN})$ and $|\mathcal{A}(T)| = \mathcal{O}(2^{KN})$, where M is the total number of modules, and N is the number of modules in a robot, and so Algorithm 1 runs in exponential time.

Algorithm 1 Find the optimal robust team with K robots

OptRobust(K)

1: $\mathcal{T}_K \leftarrow$ GenerateTeams($\mathcal{M}, K$)
2: **for all** $T \in \mathcal{T}_K$ **do**
3: Score(T) $\leftarrow \mathbb{P}(\mathbb{S}(T) \geq u_{\text{thresh}})$
4: **for all** $T \in \mathcal{T}_K$ **do**
5: $\rho(T, u_{\text{thresh}}) \leftarrow \sum_{T' \in \mathcal{A}(T)} \mathbb{P}(T', T)$Score($T'$)
6: $T^* \leftarrow \text{argmax}_{T \in \mathcal{T}_K} \rho(T, u_{\text{thresh}})$
7: **return** T^*

Since finding the optimal robust team takes exponential time, we contribute *ApproxRobust*, an algorithm that approximates the optimal robust team of size K and runs in polynomial time. Simulated annealing is performed to limit the number of teams considered; instead of considering all possible teams of size K, only $k_{\max}$ iterations of simulated annealing is run. The size of $k_{\max}$ should be sufficient to properly explore the space of robots.

The function ApproxScore(T) is used to approximates the score of a team T, by using the optimality function (e.g., ρ for risk-adverse optimality), and only considers 3 cases: the function assumes that either none of the modules in a robot team fail, a fixed number N_{fail} of the modules fail,

	No penalty		Fall-back penalty	
# of robots	OptRobust	Average team	OptRobust	Average team
1	0.577 ± 0.251	0.272 ± 0.181	0.249 ± 0.169	0.067 ± 0.052
2	0.623 ± 0.307	0.248 ± 0.231	0.134 ± 0.179	0.008 ± 0.016
3	0.653 ± 0.345	0.235 ± 0.273	0.103 ± 0.182	0.002 ± 0.006
4	0.672 ± 0.365	0.231 ± 0.300	0.089 ± 0.187	0.001 ± 0.003

TABLE I: The optimal robustness scores of teams with 1 robot (3 modules) to 4 robots (12 modules).

or all of them fail (selecting a value for N_{fail} is dependent on the domain and module failure probabilities). Hence, only $\binom{NK}{N_{\text{fail}}} + 2$ combinations of teams are considered in $\mathcal{A}'(T) \subset \mathcal{A}(T)$ and significantly reduces the computational time, albeit for an approximation of the true team score:

$$\text{ApproxScore}(T) \leftarrow \frac{1}{\eta} \sum_{T' \in \mathcal{A}'(T)} \mathbb{P}(T', T)\mathbb{P}(\mathbb{S}(T) \geq u_{\text{thresh}})$$

where $\eta = \sum_{T' \in \mathcal{A}'(T)} \mathbb{P}(T')$, so $\frac{1}{\eta}$ is a normalizing factor.

VI. Experiments and Results

We evaluate our model and algorithms using synthetic data and data from real robot experiments. We use the risk-adverse optimality below, and we believe our results and analysis will be similar for risk-controlled optimality.

A. Evaluating the ρ-SGraCR Model

In our first set of experiments, we use our ρ-SGraCR model to evaluate if our team performance model sufficiently solves the robust team formation problem. To do so, we generated 100 random instances of the ρ-SGraCR model, using 3 types of modules with 3 modules each, with randomly created module capabilities and compatibility.

We used two different settings to generate the module capabilities, *no penalty* and *fall-back penalty*. In the *no penalty* setting, all 9 module capability means were uniformly sampled from $[50, 150]$ and variances were uniformly sampled from $[0, 100^2]$. In the *fall-back penalty* setting, the 3 fall-back modules (1 for each type) had means set to 0 and variances set to 100^2, while other modules had means and variances sampled from $[50, 150]$ and $[0, 100^2]$ as before. The *fall-back penalty* setting assigns the lowest capability to fall-back modules, while the *no penalty* setting has no penalties on any modules. We were interested to see if the settings would affect the optimal robust team found.

We ran the *OptRobust* team formation algorithm to find the optimal robust team, from a size of 1 robot (3 modules) to 4 robots (12 modules). Table I shows the scores of the optimal teams of a fixed size. In the *no penalty* setting, increasing the number of robots in the team generally improves its robustness (the probability of at least attaining the performance threshold), while in the *fall-back penalty* setting, increasing the number of robots in the team generally decreases its robustness. The results are interesting as it shows that increasing *redundancy* does not always improve *robustness*. Increasing the number of robots increases the likelihood that some of them are functional, but also increases the probability that *some* module(s) will fail and lower the team performance. As such, in the *fall-back penalty* setting, this causes the overall team score to decrease.

	No penalty		Fall-back penalty	
# of robots	ApproxRobust	Difference to optimal	ApproxRobust	Difference to optimal
1	0.565 ± 0.258	0.012 ± 0.030	0.239 ± 0.171	0.011 ± 0.021
2	0.543 ± 0.322	0.080 ± 0.103	0.109 ± 0.161	0.026 ± 0.039
3	0.556 ± 0.360	0.098 ± 0.138	0.067 ± 0.132	0.036 ± 0.077
4	0.531 ± 0.386	0.141 ± 0.190	0.042 ± 0.107	0.047 ± 0.104

TABLE II: The robustness scores of teams formed by *ApproxRobust* compared to the optimal team.

Thus, the ρ-SGraCR model is capable of modeling instances where the redundancy does (in the *no penalty* setting) or does not (in the *fall-back penalty* setting), or somewhere in between (where fall-back modules have a fraction of the other modules' capabilities), demonstrating its expressiveness in the space of robust team formation problems.

B. Comparing the OptRobust and ApproxRobust Algorithms

Our second set of experiments compares the two robust team formation algorithms, *OptRobust* and *ApproxRobust*. These experiments analyze how well the approximation algorithm performs compared to the optimal team. Similar to the previous experiments, we generated 100 random instances of ρ-SGraCR models, with 3 module types of 3 modules each in the *no penalty* and *fall-back penalty* settings.

We used both robust team formation algorithms to compose a multi-robot team of 1 to 4 robots. For example, there are 3654 possible 3-robot teams, and *ApproxRobust* only searched 1000 teams with simulated annealing, so less than $\frac{1}{3}$ of the space was considered. Further, as an approximation, *ApproxRobust* assumes that all modules do not fail, $N_{\text{fail}} = 4$ fail, or all fail, and does not consider cases in between.

Table II shows the results of our experiments. The score (the probability of at least attaining a performance threshold) of the optimal team is higher than that of the team found by *ApproxRobust*, but the difference between the scores is small, which shows that *ApproxRobust* performs extremely well considering its approximations and its lower runtime.

C. Application on Real Robots

We applied our ρ-SGraCR model to real robots in the foraging domain, to demonstrate its efficacy and relevance to real robot scenarios. We used three types of robot platforms: Lego NXTs, CreBots, and Aldebaran NAO humanoid robots. The CreBots are iRobot Creates with TurtleBot hardware running our CoBot software. We chose these robots as they represent a spectrum from being easily reconfigurable (NXT) to being difficult to reconfigure (NAO).

1) The Foraging Task: The task of the multi-robot team was to forage wooden blocks to two stockpiles. Fig. 1 shows the setup of the experiment. There were 9 wooden blocks (resources to forage) in total, belonging to two types: colored (i.e., yellow, blue, and orange) and uncolored (i.e., regular brown). One of the blocks was located inside a small tunnel that was accessible only by NXTs, and two of the blocks were placed on top of the tunnel and released only when a CreBot or NAO was nearby. We set up these three blocks as "bonus" resources that can be foraged only when the right robot is included in the team.

The stockpiles on the left side of the field was for uncolored blocks, and the right for colored blocks. The robots

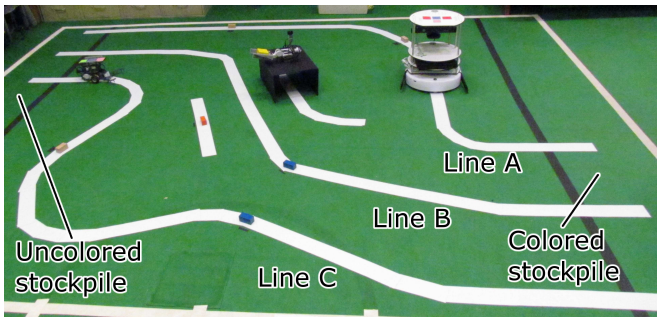

Fig. 1: The setup of the foraging experiment showing the initial robot positions and wooden block positions. Uncolored and colored wooden blocks are to be foraged to their respective stockpiles on the left and right sides of the field.

had three minutes to complete the task, and their utility was:

$$\text{Utility} = U_\text{g} \cdot (|B_\text{g,c}| + |B_\text{g,i}|) + U_\text{d} \cdot (|B_\text{d,c}| + |B_\text{d,i}|) + \sum_{b \in B_\text{g,c}} U_\text{t}(t_\text{total} - t_b) + \sum_{b \in B_\text{d,c}} U_\text{t}(t_\text{total} + t_\text{drop} - t_b) \quad (1)$$

where U_g and U_d are the utilities for foraging blocks on the ground and blocks that were dropped respectively, and U_t converts time in seconds into utilities. The first subscript (g/d) of B indicates the initial position of the block (ground, dropped), and the second subscript (c/i) indicates if the block was foraged to the correct stockpile (correct, incorrect). Blocks foraged to the correct stockpile received a time bonus based on the time remaining when the block was foraged.

2) Robot Types and Behaviors: We used three robot platforms (NXT, CreBot and NAO), and defined the modules as follows. $M_1 = \{\text{normal}^*, \text{fast}\}$ were the motors, where all platforms could use their normal motors, and the NXT had the option of faster motors. $M_2 = \{\text{no comm}^*, \text{comm}\}$ was the communication modules, that allowed the robots to communicate and coordinate. $M_3 = \{\text{no localization}^*, \text{global localization}\}$ was the localization module, that allowed the NXTs and CreBots to know their global (x, y) position (the NAOs could not have the global localization module). Lastly, to differentiate the robot platforms, $M_4 = \{\text{none}^*, \text{NXT}, \text{CreBot}, \text{NAO}\}$, where none indicated a failed/non-existent robot (explained later). The superscript * indicates that a module is the fall-back module of its type, i.e., it has a success probability of 1, and if another module fails, the robot uses the fall-back module.

Only two robots ($4 \times 2 = 8$ modules) performed the task at each trial. Robot teams are able to communicate only if *both* of them have communication modules.

The behaviors of the robots depend greatly on the configuration of the team. Generally, the NXTs would perform line following to forage the blocks on the lines connecting the two stockpiles (A, B, and C in Fig. 1), and would also forage the blocks at the disconnected lines if they had global localization. The CreBot and NAO robots would drop the blocks at the start of the trial; if they could communicate with the NXT, then they would coordinate with the NXT to maximize t_drop in Equation 1. The CreBot would forage the block closest to its initial location, and other blocks if it had global localization; the NAO does not forage any blocks. The robots did not know which blocks were colored or uncolored, unless a NAO was on the team and both robots had communication modules.

The learning of the ρ-SGraCR model and the formation of the robust team does not require any information about the behaviors of the robots — only the module success probabilities and observations of the team utilities are necessary. If the behaviors were changed, the teams would attain different utilities and a new ρ-SGraCR model would be learned.

3) Experimental Setup: Since it is difficult to get robot modules to fail on demand at the desired failure rate, we instead ran the foraging trials assuming modules were always successful, and did the analysis of module failures separately. With the modules defined in the previous subsection, there were 14 unique robot configurations (8 NXTs, 4 CreBots, and 2 NAOs) with 84 feasible two-robot teams and 8 feasible one-robot teams. Teams were considered feasible if there was at least 1 NXT in the team. The goal was to form a robust two-robot team, but a two-robot team can become a one-robot team if the robot base module fails on one of them.

We performed 30 trials in the foraging experiments to enumerate all the feasible teams. Only 30 trials were necessary since the robot behaviors did not always change based on the module configuration, e.g., a team with both robots having no communication modules performs identically to a team where one robot has the communication module.

We then performed 10-fold cross validation, where 90% of the training data (utilities of teams) was used to learn a ρ-SGraCR model, and the learned model is used to form the risk-adverse team. We set the performance threshold to be 700 for the trials, which is slightly less than the mean utility attained by the teams. We used the *ApproxRobust* team formation algorithm, where $N_\text{fail} = 4$. We only used *ApproxRobust* as the optimal algorithm *OptRobust* is infeasible to be run in general problems due to its exponential runtime.

To compare the performance of our model and algorithm, we used two benchmarks. First, we used a *highest utility heuristic*, that computed the robustness score of the team that attained the highest utility (i.e., the team that had the best utility assuming no modules failed). Second, we used a *market-based technique* where each module bid using the utility attained from the training data and module failure probabilities:

$$\text{Bid}(m) = \sum_{T \text{ s.t. } m \in T} \frac{1}{\eta} \mathbb{P}(T) \cdot \text{Utility}(T)$$

where $\frac{1}{\eta}$ is a normalizing factor.

4) Results and Analysis: Fig. 2 shows the results of our robot experiments, where the dark blue line indicates the median, the top and bottom of the box represent the 75^{th} and 25^{th} percentiles respectively, and the top and bottom whiskers represent the maximum and minimum values. The robustness scores of all the teams were distributed between 0 (worst team) and 0.64 (optimal team), with a median of 0.06 (the 25th percentile and 0th percentile are equal, so there is no bottom whisker for the distribution of all the teams).

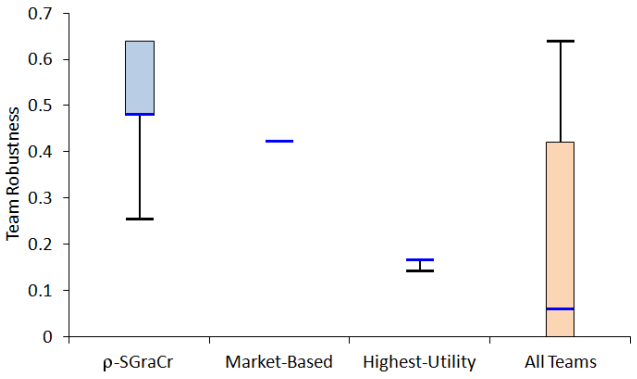

Fig. 2: The robustness scores of teams formed by ρ-SGraCR and competing approaches. The dark blue line indicates the median, the top and bottom of the box represent the 75^{th} and 25^{th} percentiles, and the top and bottom whiskers represent the maximum and minimum values.

ρ-SGraCR formed a team with a median robustness of 0.48, with the 25^{th} percentile also at 0.48. The 75^{th} percentile and maximum value are 0.64, which is the robustness of the optimal team of this experiment (and hence ρ-SGraCR has no top whisker in Fig. 2); ρ-SGraCR found the optimal team in 4 of the 10 trials. The market-based algorithm always found the same team that had a robustness score of 0.43 (and so the market-based algorithm has no box or whiskers). ρ-SGraCR outperformed the market-based algorithm in 8 of the 10 trials (the other 2 trials had robustness scores of 0.42 and 0.26). The highest-utility heuristic, i.e., picking the team that attained the highest utility, had a score of 0.17 in 9 of the 10 trials and 0.14 in 1 trial (and hence there is no box for the highest-utility heuristic).

Thus, ρ-SGraCR outperforms the competing market-based algorithm and highest-utility heuristic (with p-values of 0.025 and 0.000005 respectively using a one-tailed paired Student's t-test), and found the optimal robust team 40% of the time. In 9 of the 10 trials, the team formed is above the $75th$ percentile, which reflects that the ρ-SGraCR model effectively modeled the team performance and formed good robust teams. As such, our results demonstrate that ρ-SGraCR is well-suited to model the performance of robust multi-robot teams in challenging scenarios where the optimal team is difficult to compute *a priori*.

VII. Conclusions

Forming an effective multi-robot team to perform a complex task is difficult, especially when robots may experience failure. We defined the robust team formation problem, where robots are configured by selecting modules, and each module has an independent success rate. The performance of the multi-robot team then depends on the team's capability (the utility the team receives if all the modules function) and the probability of failure. We introduced two optimality criteria for the robust team formation problem, risk-averse optimality and risk-controlled optimality.

We contributed the Robust Synergy Graph for Configurable Robots (ρ-SGraCR) model, that is an extension of the SGraCR model we introduced recently. The ρ-SGraCR models the synergy between modules in the same robot and across robots, as well as the module success rates, and is used to compute the performance of a multi-robot team (a Gaussian mixture model). To form robust teams, we contributed two robust team formation algorithms, *OptRobust* and *ApproxRobust*, that computes the optimal robust multi-robot team in exponential time, and approximates the optimal team in polynomial time respectively.

We evaluated the ρ-SGraCR model and algorithms in a series of experiments. First, we showed that depending on the problem, the robustness of a multi-robot may or may not increase as the number of robots on the team increases, and demonstrated that the ρ-SGraCR effectively models both scenarios. We compared the *OptRobust* and *ApproxRobust* algorithms, and showed the *ApproxRobust* finds effective teams using its assumptions, with significantly lower runtime. In our robot experiments, we used three robot platforms (Lego NXT, CreBot, Aldebaran NAO) in a foraging task, and learned the ρ-SGraCR model using real robot data, and formed robust teams (finding the optimal team 40% of the time), outperforming competing approaches.

The ρ-SGraCR effectively models multi-robot team performance, taking into account module failure rates, and forms robust multi-robot teams in challenging robot domains. The ρ-SGraCR is applicable to hardware modules (e.g., motors, sensors) and software modules (e.g., selecting an appropriate algorithm for a task) of configurable robots.

Acknowledgments

The authors thank Brian Coltin for his help with the CreBots. This work was partially supported by the Air Force Research Laboratory under grant no. FA87501020165, by the Office of Naval Research under grant number N00014-09-1-1031, and the Agency for Science, Technology, and Research (A*STAR), Singapore. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

References

[1] S. Liemhetcharat and M. Veloso, "Synergy graphs for configuring robot team members," in *Proc. AAMAS*, pp. 111–118, 2013.
[2] T. Service and J. Adams, "Coalition formation for task allocation: theory and algorithms," *JAAMAS*, vol. 22, pp. 225–248, 2011.
[3] J. Chen and D. Sun, "Resource Constrained Multirobot Task Allocation Based on Leader-Follower Coalition Methodology," *IJRR*, vol. 30, no. 12, pp. 1423–1434, 2011.
[4] Y. Zhang and L. Parker, "Task Allocation with Executable Coalitions in Multirobot Tasks," in *Proc. ICRA*, 2012, pp. 3307–3314.
[5] T. Preisler and W. Renz, "Scalability and robustness analysis of a multi-agent based self-healing resource-flow system," in *Proc. FedCSIS*, 2012, pp. 1216–1268.
[6] G. Kaminka and M. Tambe, "Robust agent teams via socially-attentive monitoring," *JAIR*, vol. 12, pp. 105–147, 2000.
[7] A. Cunningham, K. Wurm, W. Burgard, and F. Dellaert, "Fully distributed scalable smoothing and mapping with robust multi-robot data association," in *Proc. ICRA*, 2012, pp. 1093–1100.
[8] N. Napp and E. Klavins, "Robust by composition: Programs for multi-robot systems," in *Proc. ICRA*, 2010, pp. 2459–2466.
[9] S. Stein, E. Gerding, A. Rogers, K. Larson, and N. Jennings, "Algorithms and Mechanisms for Procuring Services with Uncertain Durations using Redundancy," *AIJ*, vol. 175, no. 14-15, 2011.