2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Approximated Dynamic Trait Models for Heterogeneous Multi-Robot Teams

Glen Neville†, Harish Ravichandar†, Kenneth Shaw† and Sonia Chernova†

Abstract— **To realize effective heterogeneous multi-agent teams, we must be able to leverage individual agents' relative strengths. Recent work has addressed this challenge by introducing trait-based task assignment approaches that exploit the agents' relative advantages. These approaches, however, assume that the agents' traits remain static. Indeed, in real-world scenarios, traits are likely to vary as agents execute tasks. In this paper, we present a transformation-based modeling framework to bridge the gap between state-of-the-art task assignment algorithms and the reality of dynamic traits. We define a transformation as a function that approximates dynamic traits with static traits based on a specific statistical measure. We define different candidate transformations, investigate their effects on different dynamic trait models, and the resulting task performance. Further, we propose a variance-based transformation as a general solution that approximates a variety of dynamic models, eliminating the need for hand specification. Finally, we demonstrate the benefits of reasoning about dynamic traits both in simulation and in a physical experiment involving the game of capture-the-flag.**

I. INTRODUCTION

Heterogeneous multi-agent systems offer the potential to solve complex problems in various domains, such as agriculture [1], military [2], assembly [3], and warehouse automation [4], that would otherwise be infeasible for a single agent. To solve such complex problems, we require accurate models of agent capabilities that help teams leverage the relative strengths of the individual agents effectively.

Modeling agent capabilities is an inherent, but often overlooked, part of heterogeneous multi-agent coordination. In multi-agent systems, the framework used for modeling agents and tasks is crucial to defining the types of problems that can be solved and the methods available for solving them. Although not often explicitly discussed, multi-agent coordination approaches make implicit assumptions about how traits are modeled. For instance, agents' abilities have been modeled as constant-valued utilities that indicate whether an agent either can or can not perform a task (e.g., [4]).

Accurately modeling traits requires an understanding of how traits evolve and change dynamically with respect to time or other relevant variables. For instance, as a robot performs a task or as time evolves, traits such as payload or speed may change due to changes in battery levels. However, modern algorithms for multi-agent coordination are designed to treat all traits as static and do not consider dynamic traits (e.g., [5], [6], [7]).

In this work, we enable modern multi-agent coordination approaches to reason about dynamic traits. Specifically, we present a transformation-based framework for approximate modeling of dynamic traits. Through these transformations, we enable multi-agent coordination algorithms to extract the most relevant information from dynamic models to make informed decisions.

We define a *transform* as a function that approximates a dynamic trait model with a static trait model. The transforms introduced in this paper work to strategically select static values that adequately compensate for not being able to directly reason about dynamic traits. We introduce and evaluate a variety of transforms to approximate different dynamic trait models as static trait models. These transforms are based on statistical measures, such as the mean and minimum trait values over a pre-specified operating range.

We find that different transformations are more suited for different dynamic traits. As a result, a key challenge is choosing the transform that best approximates a given dynamic trait model. To address this challenge, we also introduce a variance-based transform. This variance-based transformation seeks to identify the point on the curve that exhibits the highest variance across all team members. Thus, the high variance transform highlights the differences between the robots, allowing coordination algorithms to better reason about an agents' strengths and weaknesses and make the most impactful decisions.

To illustrate our approach's effectiveness, we combine our proposed representation with a state-of-the-art multi-agent task allocation (MRTA) algorithm [5]. We evaluate the benefits of modeling and approximating dynamic traits using a simulated and a physical capture-the-flag game environment. Our results show that using the proposed transforms leads to statistically significant improvements in task performance over techniques that consider only static traits. Further, we were able to show that the high-variance transform outperformed all other transforms across several dynamic models.

In summary, our contributions include: i) creating a modeling framework that allows modern multi-agent algorithms to reason about dynamic traits, ii) demonstrating that reasoning about dynamic traits can improve task performance of modern multi-agent algorithms, iii) showing that the particular transformation used to reason about dynamic traits can significantly impact task performance, and iv) introducing a general solution using a high variance method of

*This work was supported by the Army Research Lab under Grant W911NF-17-2-0181 (DCIST CRA)

†The authors are with the Institute of Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA, USA {gneville, harish.ravichandar, kshaw, chernova}@gatech.edu

transformation that performs well over a variety of dynamic traits and outperforms several other transformations.

II. Related Work

The first and most straightforward way of modeling agents is to avoid modeling agents altogether and model tasks as a list of the number of each agent of each type required to complete the task [8]. Unfortunately, this method is not flexible to the addition of new agent types or different team compositions for a task.

To address this, some frameworks model agents by a list of utilities that describe how well each agent would perform each task [9], [10], [11]. This allows these systems to be extensible to new agent types easily, but it does not allow multiple robots to work collaboratively on a single task.

Other frameworks model agent capabilities a list of available traits, each defined either by binary continuous variables [5], [6]. Such models are paired with a model of tasks that denotes a task by the combination and amounts of specific traits needed to complete said task.

There are works that model capabilities at an even lower level by generating abilities by analysing the agents sensors and actuators directly and constructing list [12]. The benefit to these systems is that they can responded to certain types of change in an agents traits (e.g. the lose of a sensor) however they often do not model less pronounced changes in sensor quality or actuator performance.

Each of the above methods allows for multi-agent algorithms to understand the relative strengths and weaknesses of various agents. However, none of these modeling frameworks allow for traits to change dynamically, which is an issue since an agent's traits often change throughout execution. These dynamically changing traits can be caused by various factors, including battery levels, actuator degradation, or, in the case of human agents, fatigue. In contrast to existing work our work allows existing algorithms to consider how traits change during execution.

To accomplish this we take inspiration from the human factors community that often model human capabilities with models that inherently capture the dynamic nature of human performance. Works in this space often model performance as a mathematical function of a set of hyperparameters and have been shown of being able to model: human running human athletic models [13], [14], fatigue [15] and human mental performance [16]. We use a similar mathematical form to model robot traits and through our transformation framework allow existing algorithms to reason about these models.

III. Task Assignment

To test our framework, we use it alongside a task allocation algorithm to show how modeling traits can improve performance. The task assignment problem addresses how to assign agents to tasks best to maximize overall team performance. Research in this area has been characterized by three factors [11], [17]. The first factor is task type and falls into two categories single-robot and multi-robot. These categories describe the number of robots required to complete each task in the system. The second factor is the robot type and is split between single-task and multi-task and describes the number of tasks a robot can perform simultaneously. The final factor is the assignment type, and it is split between instantaneous and time-extended. These categories define whether the task allocation algorithm considers only the current task performance versus future scheduling constraints.

To validate our framework, we show our framework's performance paired with a Single-Task Robots Multi-Robot Tasks Instantaneous Assignment (ST-MR-IA) algorithm. We refer readers to [11], [17] for a comprehensive explanation of all categories of task assignment and pertinent examples. Previous researchers have proposed several heuristic-based solutions to the ST-MR-IA problem [11], [18]. The authors of [19] present a distributed set-covering algorithm to solve the MT-MR-IA problem. Several auction-based solutions to the MTRA problem have been shown in the recent research [20], [21], [7]. The issue with most existing approaches to tackling this problem is that the desired team structure is specified only in terms of an ideal distribution of agents. There are, however, two notable exceptions to reasoning only about distributions of agents in the research. These are the frameworks introduced in [6] and [5], both capable of receiving the role requirements provided in the form of a desired trait distribution across roles.

The authors of [5] introduce a framework called STRATA (Stochastic TRAit-based Task Assignment), which uses a method similar to [6], and does not assume that the desired distribution of agents is known. Our work builds on [5] and [6] by expanding the types of traits about which these algorithms can reason. Due to its flexibility in what types of traits it allows, we use STRATA in conjunction with our framework. In the interest of completeness, we provide a brief introduction to STRATA below (refer to [5] for an in-depth treatment).

STRATA assumes that each agent in the team belongs to a particular species. To model the effects of task assignments and team capabilities, STRATA defines the following three variables:

- *Species-Trait distribution*: The traits of a heterogeneous team are summarized in the species-trait distribution matrix $Q \in \mathbb{R}^{S \times U}$, where S is the number of species and U is the number of traits. The ijth element of Q quantifies the jth trait of the ith species.
- *Task-Species distribution*: Strata denotes Task assignments by the task-species distribution matrix $X \in \mathbb{N}^{M \times S}$, where M is the number of tasks. The task-species distribution matrix X defines which agents are assigned to each task.
- *Task-Trait distribution*: The effects of task assignments (X) and the species' traits (Q) on how the traits are aggregated for each task are summarized in the task-trait distribution matrix $Y \in \mathbb{R}^{M \times U}$ and is given by $Y = XQ$.

STRATA performs task assignment by solving an optimization problem that seeks to minimize the difference between the task-trait distribution of the team (Y) and a specified desired distribution (Y^*). STRATA allows for the optimization of two different types of task assignment: *exact matching* and *minimum matching*. Exact matching seeks to minimize the difference between achieved trait distribution and desired trait distribution. On the other hand, minimum matching does not penalize over-provisioning and aims to prevent trait distribution from being less than the desired wherever possible.

Our framework seeks to extend the capabilities of algorithms like STRATA, allowing them to reason about dynamic traits that change throughout execution.

IV. Modeling Dynamic Traits

A. Dynamic Trait Transformation

As seen in the preceding section, modern solutions to the MRTA problem assume that the agents' capabilities remain unchanged and rely on static trait models. However, traits do not often remain static and are instead defined by functional models that describe their dynamic nature. For instance, a robot's velocity v could be modeled as $v(b) = S_{max} * b$, where $b \in [0, 1]$ denotes the battery level and $S_{\max}$ is the maximum speed. To bridge the gap between the reality of dynamic traits and the practical limits of existing task assignment frameworks, we present the following framework for systematically approximating dynamic trait functions into static trait values.

In our framework, we allow for the proper treatment of both static and dynamic traits. We define a set of N_s static traits as $\mathcal{S}$, with its ith element $s_i \in \mathbb{R}$ describing a particular static trait. We also define a set of N_d dynamic traits as $\mathcal{D}$, with its ith element describing dynamic trait function $d_i : \mathcal{X}_i \to \mathbb{R}$, where $\mathcal{X}_i \subseteq \mathbb{R}$ denotes the domain of $d_i(\cdot)$ and $\mathcal{F}$ denotes the space of algebraic and transcendental functions such that $d_i \in \mathcal{F}$.

In order to reason over the dynamic traits using existing task assignment frameworks, we consider a set of transformations that approximate dynamic functions. The set of N_τ transformations is given by $\mathcal{T}$ with each element defined as a transformation $\tau_j : \mathcal{D} \times \Theta_j \to \mathcal{S}'$ that maps a dynamic trait into a static trait. $\mathcal{S}'$ is the set of N_d static traits (each denoted by $s'_i \in \mathbb{R}$) formed as a result of transforming the set of dynamic traits $\mathcal{D}$. Θ_j is the space of hyperparameters associated with τ_j, and N_τ is number of transformations. See Fig. 1 for more details.

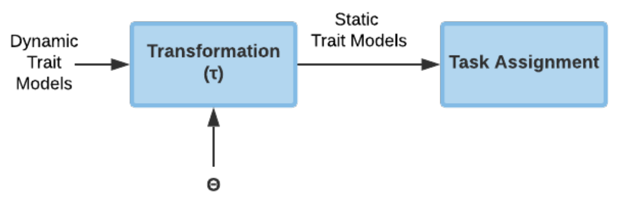

Fig. 1: Proposed pipeline between presented framework and task assignment

For instance, consider the transformation $\tau_{\min}$, which returns the global minimum over a specified range. Applying this transformation to a dynamic trait model $d_i(\cdot)$ can be denoted by $s'_i = \tau_{\min}(d_i, \theta_{\min})$, where s'_i is the transformed trait which contains the global minimum and $\theta_{\min} = [a, b]$ contains the hyper-parameters that specify the range of $x \in \mathcal{X}_i$ to be considered.

B. Simple Dynamic Model Transforms

In this work, we validate the following transforms for dynamic models:

- $\tau_{\max}(d_i, \theta_{\max})$, which returns the maximum value of $d_i(\cdot)$ over the range of x defined by the hyper-parameters $\theta_{\max} = [a, b]$.
- $\tau_{\min}(d_i, \theta_{\min})$, which returns the minimum value of $d_i(\cdot)$ over the range of x defined by the hyper-parameters $\theta_{\min} = [a, b]$.
- $\tau_{\text{mean}}(d_i, \theta_{\text{mean}})$, which returns the average value of $d_i(\cdot)$ over the range of x defined by the hyper-parameters $\theta_{\text{mean}} = [a, b]$.
- $\tau_{3\text{Q}}(d_i, \theta_{3\text{Q}})$, which returns the value of $d_i(\cdot)$ at the third quartile position over the range of x defined by the hyper-parameters $\theta_{3\text{Q}} = [a, b]$.

C. Variance-based Dynamic Model Transforms

In addition to the above transforms, we also present two variance-based transforms to adaptively approximate a wide variety of trait models.

$\tau_{\text{HVar}}(\cdot)$: We define the first variance-based transform as

$$\tau_{\text{HVar}}(d_i, \theta_{\text{HVar}}) = d_i(x_{\max}) \tag{1}$$

where $x_{\max} = \arg\max_{x \in \theta_{\text{HVar}}} \text{var}(d_i(x))$, $\text{var}(d_i(x))$ denotes the sample variance of $d_i(x)$ computed across all species (see Fig. 2 for an example), and $\theta_{\text{HVar}} = [a, b]$. Thus, $\tau_{\text{HVar}}(\cdot)$ returns the value of $d_i(\cdot)$ evaluated at the point that maximizes the variance of the corresponding trait across all the species, thereby accentuating the differences between agents. We hypothesize that this transform outperforms other transforms as it allows the task-allocation system to understand the differences between agents better when assigning roles.

$\tau_{\text{nHVar}}(\cdot)$: We define the second variance-based transform as

$$\tau_{\text{nHVar}}(d_i, \theta_{\text{nHVar}}) = \frac{\tau_{\text{HVar}}(d_i, \theta_{\text{nHVar}})}{\frac{1}{C}\sum_{k=1}^{C} \tau_{\text{HVar}}(d_i, \theta_{\text{nHVar}})} \tag{2}$$

where $\theta_{\text{nHVar}} = [a, b]$ are the hyperparameters that define the normalization range for traits. Thus, $\tau_{\text{nHVar}}(\cdot)$ returns the value of $d_{ik}(\cdot)$ at the the value of x that maximizes the variance between all of the agents' normalized traits. We note that $\tau_{\text{nHVar}}(\cdot)$ is similar to $\tau_{\text{HVar}}(\cdot)$ in that both seek to maximize the difference between agents. However, $\tau_{\text{nHVar}}(\cdot)$ disregards information about absolute quantities and instead allows the task allocation system to reason about the relative magnitude of agents' traits. Both $\tau_{\text{HVar}}(d_{ik}, \theta_{\text{HVar}})$ and $\tau_{\text{nHVar}}(d_{ik}, \theta_{\text{nHVar}})$ have opposite low variance equivalence

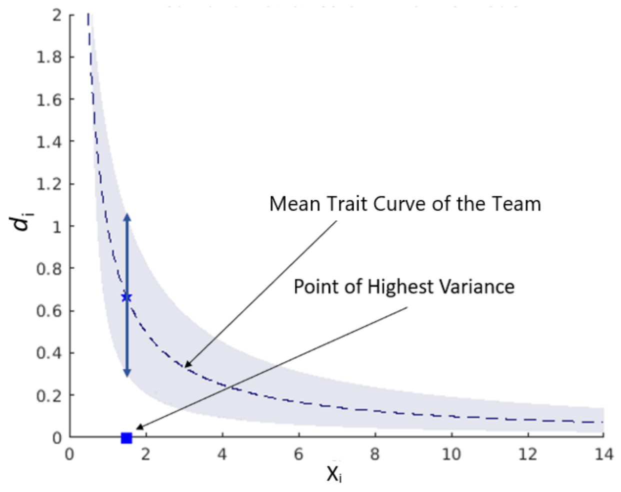

Fig. 2: Example of the mean and variance of a dynamic trait associated with an entire team composed of different species.

$\tau_{\text{LVar}}(d_{ik}, \theta_{\text{LVar}})$ and $\tau_{\text{nLVar}}(d_{ik}, \theta_{\text{nLVar}})$ which perform similarly to their high variance counterpart except they seek to minimize the variance between all of the agents' abilities.

Given the definitions of static traits $\mathcal{S}$ and transformed dynamic traits $\mathcal{S}'$, we now model the capabilities of agents within the team. As mentioned earlier, we assume that each agent in the team belongs to a particular species. We define the traits of the kth species as $q^{(k)} = [s_1(k), s_2(k), \cdots, s_{N_s}(k), s'_1(k), s'_2(k), \cdots, s'_{N_d}(k)] \in \mathbb{R}^U$, where $U = N_s + N_d$ is the total number of traits, $s_i(k)$ and $s'_j(k)$ are the ith static and jth approximated dynamic traits of the kth species, respectively. Now, the species-trait distribution matrix is given by $Q = [{q^{(1)}}^T, {q^{(2)}}^T, \cdots, {q^{(S)}}^T]^T \in \mathbb{R}^{S \times U}$, where S is number of species in the team.

V. Experimental Design

To evaluate the framework, we conducted three experiments to explore (1) how the choice of transform used for dynamic traits affects task assignment performance, (2) whether reasoning about dynamic traits improves performance over thinking solely on static traits, and (3) whether the benefits of dynamic trait reasoning observed in simulation successfully transfer to a real-world multi-robot system.

A. Task Assignment Test-bed

To run the experiments discussed above, we developed a test-bed for understanding how our modeling framework could aid task assignment algorithms. A detailed capture-the-flag game environment was created both in simulation and on the Georgia Tech Robotarium (a multi-robot testing platform) [22]. The goal of the game was to traverse a field, capture an opponent's flag, and return it to a home base before time ran out. The team that returned their opponent's flag to their base first would *win* the game, and this would be considered as a *loss* to the opponent. If neither team was able to bring their opponent's flag back to their base in the allotted time, the team that brought their opponent's flag closest to their home base was assigned a *partial win*, and this would count as a *partial loss* to the opponent. If neither team moved the flag or the teams achieved the objective at the same time, the game ended in a *draw*.

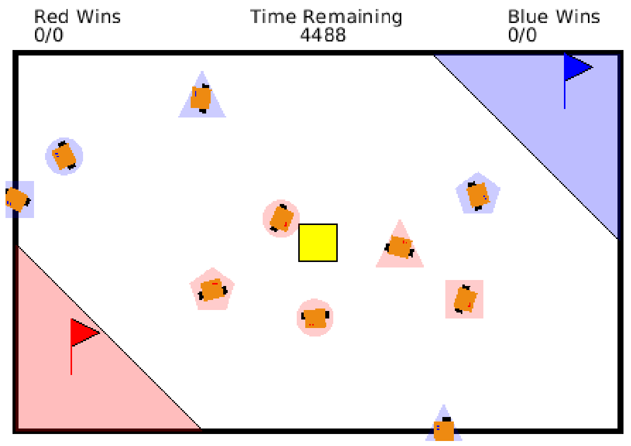

Fig. 3: Snapshot from the simulation of the capture-the-flag game with ten agents, each marked with a label indicating its role. Red Team used the proposed framework for task assignment, and Blue team's roles were randomly assigned.

On each of the teams, there were four possible roles for each agent:

- *Defender (d):* The defender's goal is to slow the opposition team's progress by protecting the flag and stunning all opponents that come within their reach. A stunned robot would be unable to move for a time equal to the difference between the stunned robot's defense score and the stunning robot's attack score.
- *Attacker (a):* The agents assigned to this role tracked down opponents holding the flag and stole the flag from them if within reach. Attackers would return any stolen flag to their base.
- *Impeder (i):* Impeders sought to position themselves between their opponents and their opponent's goal location. This role makes it more difficult for opponents to reach their goal and give their team more time to win the game.
- *Flagger (f):* The flagger role involved navigating the field to capture the opponent's flag and return it to its home base.

Each of the robots had six static traits that varied in magnitude between robots. We randomly generated each agent's traits from a uniform distribution over the ranges specified below for every game. The six static traits and their ranges are as follows

- *Carry ability:* a binary trait (0 or 1) describing a robot's ability to pick up a flag.
- *Strength:* a continuous trait, sampled in the range $[0.1, 1.2]$, that described how far an agent was able to carry the flag before dropping the flag.
- *Attack:* a continuous trait, sampled from the range $[0, 1.2]$, that described the agent's ability to stun opponents.
- *Defense:* a continuous trait, sampled from the range $[0, 1.5]$, that described the agent's ability to resist being

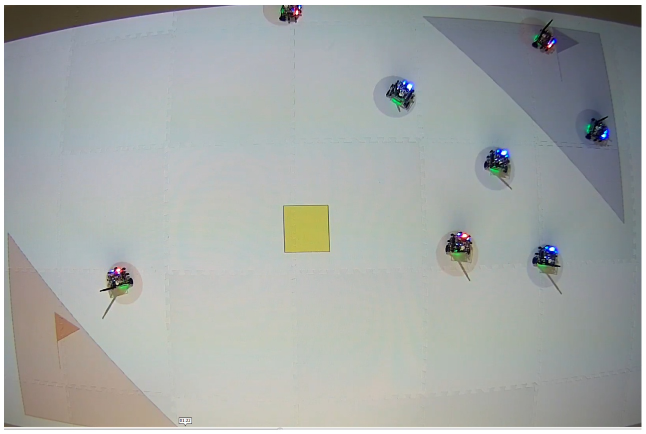

Fig. 4: We demonstrate our approach on a physical implementation of Capture-the-flag.

stunned by an opponent. If a robot's defense was higher than an attacking robot's attack, it could avoid being stunned.

- *Viewing Distance:* a continuous trait, sampled from the range $[0.15, 0.75]$, that described the maximum distance a robot could see and thus locate the flag and other robots.
- *Reach:* a continuous trait, sampled from the range $[0.5, 0.95]$, that described the maximum distance between a robot and i) its opponent, to be able to stun or steal the flag; ii) its home base, to be able to return the flag.

B. Dynamic Traits

In addition to the static traits listed above, we defined *speed* as a dynamic trait function. The speed trait sets the maximum speed at which a robot can move. We examined three different dynamic models for speed. The first dynamic model, based on human athletic models [23], is given by

$$v_e(y) = \frac{y}{\alpha y^{\beta}} \tag{3}$$

where v_e represents the agents maximum speed, y represents the distance already traveled by the agent, and $\alpha \in [1.0, 1.1]$ and $\beta \in [1.0, 2.0]$ are constants specific to individual agents.

The second dynamic model of speed is given by

$$v_h(y) = a(1 - H(y - b)) \tag{4}$$

where v_h represents the agents maximum speed, y represents the distance already traveled by the agent, the function $H(\cdot)$ is the Heaviside step function, and $a \in [0.8, 2.0]$ and $b \in [5.0, 14.0]$ represent constants that are a function of the individual robot

The third dynamic model is a linear model given by

$$v_l(y) = m * y + c \tag{5}$$

where v_l represents the agents' maximum speed, y represents the distance already traveled by the agent, and $c \in [0.6, 1.5]$ and $m \in [-0.04, -0.22]$ represent constants that are a function of the individual robot. Like the static traits, for each game and agent, the dynamic trait functions' hyper-parameters were uniformly sampled from their respective ranges.

VI. Effects of Transform on Task Assignment Performance

In this section we report the effects of different transformations on each dynamic model.

A. Experimental Setup

For each of the transformations described above and two low variance transformations, we randomly generated 200 teams for each of the transformations. We transformed each agent's speed trait in each of these teams using the framework, and we utilized STRATA to perform task assignments while only considering the transformed dynamic trait. Based on the computed task assignment, each team played a simulated round of capture-the-flag against a team with identical traits with randomly assigned roles. We repeated this process for each of the tested dynamic models and recorded the results of the games. Finally, as a baseline to the test performed above, we ran a randomly assigned team through a set of 200 games against another team with randomly assigned roles.

B. Results

Figure 5 presents the results of evaluating the eight different transforms in the simulated capture-the-flag game discussed above. For each of the transforms, a score metric was calculated based on the results of the games. The score was calculated by giving one point for a win, a half-point for a partial win, a negative half-point for a partial loss, and a negative point for a loss. We observe that the τ_{HVar} transform leads to the best performance, outperforming other transforms regardless of the dynamic trait model. We also find that τ_{LVar} consistently performed poorly, effectively hindering task assignment and leading to a performance below that of random. To show the effectiveness of the high-variance method, we also ran a pair-wise chi-squared analysis on the raw data between the high-variance transform and each of the other transforms. A chi-squares analysis was chosen as it is capable of determining if there is a statistical significant difference in the number of wins, loses, partial wins and partial loses between the different allocation methods. In 5 significant results, each of the chi-squares is denoted with an asterisk.

As can be seen from the results, the high-variance transformation outperformed all other transforms across all three models. This increased performance is likely due to the high-variance transform's ability to accentuate differences between agents. To explain, by identifying where the dynamic traits associated with different species vary the most, the high-variance transform computes approximations that are the most influential in task assignment. While the results demonstrate a statistically significant benefit on the presented task when reasoning about dynamic traits, we note that the relative importance of dynamic traits could vary from task to task.

It can also be seen that, in the case of each model, a different transform performed similarly to the high-variance

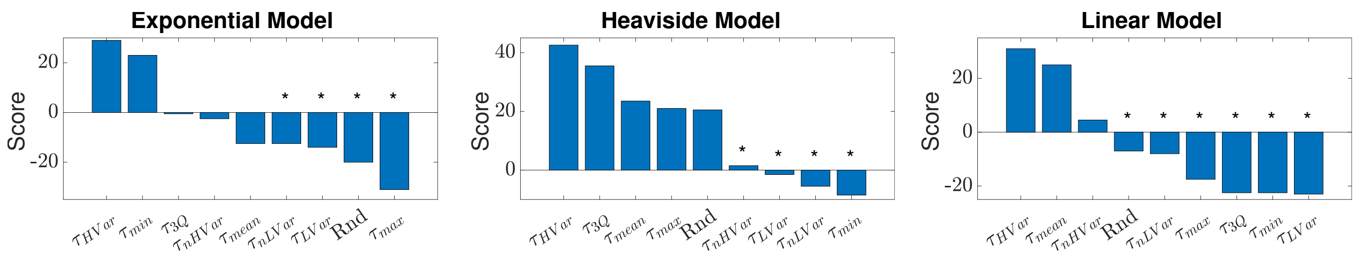

Fig. 5: Aggregate score performance against random task assignment when utilizing each transformation for different ground truth dynamic trait models. Starred columns indicate statistically significant difference when compared to the high-variance transformation, τ_{HVar}.

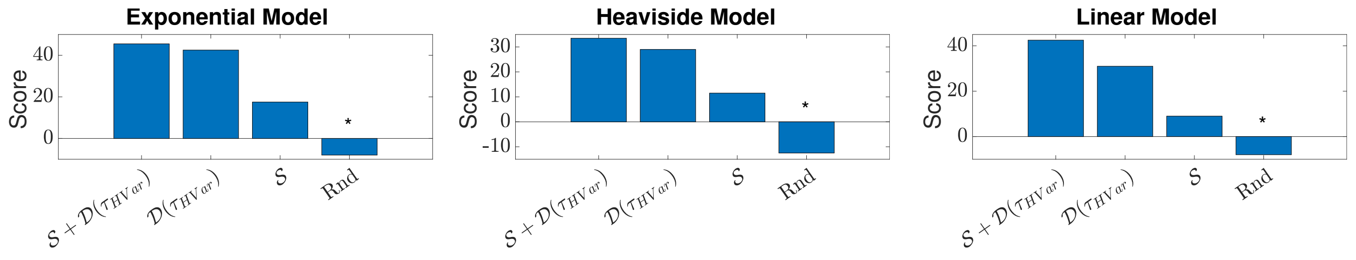

Fig. 6: Aggregate score performance against random task assignment when reasoning about different trait sets for each ground truth dynamic trait model. Starred columns indicate statistically significant difference when compared to the set of games that reasoned about dynamic and static traits, $\mathcal{S}+\mathcal{D}(\tau_{HVar})$.

transform. For the exponential model, τ_{min} performed similarly to τ_{nHVar}. Similarities in performance can be also found between τ_{nHVar} and τ_{3Q} in the Heaviside model, and between τ_{nHVar} and τ_{mean} in the linear model. Such similarities are likely caused by the transforms computing similar approximations of the dynamic trait. For example, in the step model, the point of maximum variance is near the 3rd quartile of the curve as it represents a point at which some but not all of the models have transitioned to zero. Thus, τ_{nHVar} and τ_{3Q} likely chose similar values as approximations and subsequently resulted in a similar performance. More generally, while different transformations are appropriate for different models, τ_{nHVar} is consistently suitable for all models.

In summary, we find *i)* that the choice of transform significantly affects the quality of the underlying task assignment method and *ii)* that τ_{HVar} outperform other methods for all of the dynamic models we evaluated.

VII. Value of Dynamic Trait Reasoning

In this section we report the benefits of modeling dynamic traits over just static traits.

A. Experimental Setup

Four conditions were analyzed: reasoning solely on dynamic traits ($\mathcal{D}$), reasoning solely on static traits ($\mathcal{S}$), reasoning on both dynamic and static traits ($\mathcal{S}+\mathcal{D}$), and randomly assigned roles. Similar to the first experiment, 200 teams were randomly generated for each of the tested conditions. Teams that reasoned about dynamic traits had their traits transformed using the high-variance transformation. After this, the agents were assigned to roles using STRATA. Each team played a simulated round of capture-the-flag against an identical team with randomly assigned roles. We repeated this process for each of the tested dynamic models. We recorded the results of each of the games similarly to experiment 1.

B. Effects of Reasoning on Dynamic Traits

The results of these games are shown in Figure 6. As can be seen, reasoning about dynamic traits can improve team performance compared to randomly assigned teams and teams that only consider static traits. These results clearly show the need for frameworks that can reason about dynamic traits.

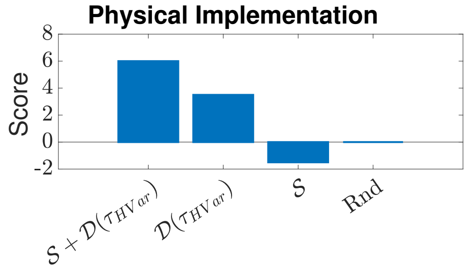

Fig. 7: Performance on the Robotarium against random task assignment when reasoning about different trait sets.

VIII. Physical Implementation

A. Experimental Setup

In this experiment, we evaluate the effectiveness of our framework on a physical multi-agent system. The physical implementation of the capture-the-flag games was designed to work with the Georgia Tech Robotarium [22] (an existing multi-robot test-bed) and followed the same structure as the simulation presented above. Using this physical implementation of the game, we evaluated the effectiveness of reasoning about exponential trait models using the τ_{HVar}

transformation. Similar to the experiments in Section VII, we constructed four conditions ($\mathcal{S}$+$\mathcal{D}$, $\mathcal{D}$, $\mathcal{S}$, and Random). We played a total of 12 games for each condition.

B. Results

This experiment looked to see how the benefits of reasoning about dynamic traits observed in the simulations from the first two experiments translate to a physical multi-robot system. As can be seen in Figure 7, scores in the physical test match those seen in simulation and confirm the benefits of reasoning about dynamic traits. Due to the importance of the dynamically changing speed trait in this task, the dynamic transform which is capable of considering this trait significantly outperforms the static transformation. By reasoning about the dynamically changing speed trait, the task allocation assigned the fastest agents to roles that could benefit from the increase in speed, improving the team's performance.

IX. Discussion

In this series of experiments, the presented framework was used alongside a time instantaneous task allocation algorithm. However in the future we hope to incorporate this system with a time-extended task allocation algorithm. Time extended task allocation considers scheduling constraints and task orderings. The presented framework would allow the algorithm to consider the dynamic nature of robot traits while performing assignments. This would enable the task allocation algorithm to consider the effects that assigning robots to earlier tasks will have on the robot's traits when assigned them to later tasks.

Finally, our framework is potentially useful for multi-agent teams consisting of both human and robotic agents. Humans-robot teams offer the opportunity to leverage the relative strengths of both groups. Human traits are often dynamic concerning factors such as fatigue, stress, or mental state (e.g., see [15] and references therein). Although not yet evaluated with human teams, our approach's ability to model dynamic traits could make it a promising mechanism to effectively allocate tasks in human-robot teams.

Although not tested in this experiment, our framework allows for multiple transformations of a single trait. We hypothesize that this could extract more information from the underlying model and further improve results.

X. CONCLUSIONS

In this paper, we have presented a novel framework for modeling the dynamic traits of heterogeneous multi-robot teams. Our framework expands the modeling capabilities available for multi-agent coordination problems. Our experimental results on simulated and physical multi-agent games suggest that reasoning about dynamic traits can lead to significant improvements in task performance. In particular, the proposed variance-based transformation of dynamic traits is shown to be consistently suitable for approximating a variety of dynamic traits.

References

[1] T. Blender, T. Buchner, B. Fernandez, B. Pichlmaier, and C. Schlegel, "Managing a Mobile Agricultural Robot Swarm for a seeding task," in *IECON Proceedings (Industrial Electronics Conference)*, 2016.

[2] C. J. McCook and J. M. Esposito, "Flocking for heterogeneous robot swarms: A military convoy scenario," in *Proceedings of the Annual Southeastern Symposium on System Theory*, 2007.

[3] L. Matthey, S. Berman, and V. Kumar, "Stochastic strategies for a swarm robotic assembly system," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2009.

[4] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," in *AI Magazine*, vol. 29, no. 1, 2008.

[5] H. Ravichandar, K. Shaw, and S. Chernova, "STRATA: A Unified Framework for Task Assignments in Large Teams of Heterogeneous Robots," *Journal of Autonomous Agents and Multi-Agent Systems*, 2019.

[6] A. Prorok, M. A. Hsieh, and V. Kumar, "The Impact of Diversity on Optimal Control Policies for Heterogeneous Robot Swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, 2017.

[7] P. M. Shiroma and M. F. Campos, "CoMutaR: A framework for multi-robot coordination and task allocation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, 2009.

[8] M. Gombolay, R. Wilcox, and J. Shah, "Fast Scheduling of Multi-Robot Teams with Temporospatial Constraints," in *EEE Transactions on Robotics, vol. 34, no. 1*, 2016.

[9] S. Vajda and D. Gale, "The Theory of Linear Economic Models," *The Mathematical Gazette*, vol. 46, no. 358, 1962.

[10] S. C. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2, 1999.

[11] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, 2004.

[12] S. A. Deloach, W. H. Oyenan, and E. T. Matson, "A capabilities-based model for adaptive organizations," *Autonomous Agents and Multi-Agent Systems*, vol. 16, no. 1, 2008.

[13] D. A. Blythe and F. J. Király, "Prediction and quantification of individual athletic performance of runners," *PLoS ONE*, vol. 11, no. 6, 2016.

[14] R. H. Morton, "Modelling human power and endurance," *Journal of Mathematical Biology*, 1990.

[15] A. H. DeCostanza, A. R. Marathe, A. Bohannon, A. W. Evans, E. T. Palazzolo, J. S. Metcalfe, and K. McDowell, "Enhancing Human–Agent Teaming with Individualized, Adaptive Technologies: A Discussion of Critical Scientific Questions," *IEEE Brain*, no. June, 2018.

[16] M. E. Jewett and R. E. Kronauer, "Interactive mathematical models of subjective alertness and cognitive throughput in humans," in *Journal of Biological Rhythms*, 1999.

[17] G. Korsah, A. Stentz, and M. Dias, "A comprehensive taxonomy for multi-robot task allocation," *International Journal of Robotics Research*, vol. 32, pp. 1495–1512, 3 2013.

[18] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Studies in Computational Intelligence*, vol. 604, 2015.

[19] O. Shehory and S. Kraus, "A kernel-oriented model for autonomous-agent coalition-formation in general environments," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1087, 1996.

[20] M. Bernardine Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," 2006.

[21] Z. Talebpour, S. Savare, and A. Martinoli, "Market-based coordination in dynamic environments based on the Hoplites framework," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-September, 2017.

[22] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 3 2017, pp. 1699–1706.

[23] D. A. Blythe and F. J. Király, "Prediction and quantification of individual athletic performance of runners," *PLoS ONE*, 2016.