A Talent-infused Policy-gradient Approach to Efficient Co-Design of Morphology and Task Allocation Behavior of Multi-Robot Systems

Prajit KrisshnaKumar[1], Steve Paul[1] and Souma Chowdhury[1,2,†]

Abstract— **Interesting and efficient collective behavior observed in multi-robot or swarm systems emerges from the individual behavior of the robots. The functional space of individual robot behaviors is in turn shaped or constrained by the robot's morphology or physical design. Thus the full potential of multi-robot systems can be realized by concurrently optimizing the morphology and behavior of individual robots, informed by the environment's feedback about their collective performance, as opposed to treating morphology and behavior choices disparately or in sequence (the classical approach). This paper presents an efficient concurrent design or co-design method to explore this potential and understand how morphology choices impact collective behavior, particularly in an MRTA problem focused on a flood response scenario, where the individual behavior is designed via graph reinforcement learning. Computational efficiency in this case is attributed to a new way of near exact decomposition of the co-design problem into a series of simpler optimization and learning problems. This is achieved through i) the identification and use of the Pareto front of Talent metrics that represent morphology-dependent robot capabilities, and ii) learning the selection of Talent best trade-offs and individual robot policy that jointly maximizes the MRTA performance. Applied to a multi-unmanned aerial vehicle flood response use case, the co-design outcomes are shown to readily outperform sequential design baselines. Significant differences in morphology and learned behavior are also observed when comparing co-designed single robot vs. co-designed multi-robot systems for similar operations.**

I. Introduction

Inspired by natural systems, Multi-Robot systems (MRS) and Swarm Systems (SS) employ collective intelligence principles to exhibit emergent behavior to accomplish tasks that are beyond the capabilities of any single robot. Emergent behavior results from simple rules followed by each entity and their interaction with each other and their environment [1]. These interactions give rise to complex adaptive behaviors that are robust and efficient. Usually such collective behavior is not readily predictable (e.g., via scaling or simple equations) from individual behavior without the use of empirical evaluations via simulations. This is because appropriate design and behavior choices at the individual robot level can lead to collective performance that is greater than the sum of its parts

Now, the physical design aka morphology of individual robots, including geometry and component choices w.r.t. sensors, actuators, computing, communication, etc., influence and constrain their operating envelope and functionalities. These design choices define the individual robot's capabilities (e.g., range, nominal power consumption, weight, sensing FoV, payload capacity, turning radius, etc.), and constrain the behavior space in which the robot can operate. On the other hand, the behavior (decision-system that perceives the environment and provides action) must align with the capabilities defined by its morphology. This creates a coupling of morphology and behavior individually. When working as a team, due to its task parallelization property, there are non-linear shifts in these constraints that affect its collective behavior. Realizing the true potential of swarm systems involves addressing formidable challenges regarding the design choices and behavior of the individual members. Even minor modifications in the design of individual robots might necessitate completely different behaviors.

A common approach to designing swarm systems is by trial and error [2]. The alternate method is the automated design approach, where the behavior is formulated as an optimization problem to be solved [3]–[8]. These methods optimize the behavior of the individual robots using evolutionary methods and Reinforcement Learning (RL) methods to find the optimal behavior of individual robots that leads to the desired collective performance [9], [10], [11], [12]. By optimizing or prescribing the morphology first (as is typical), the capability space is inherently confined without considering the behavioral space, leading to a sub-optimal emergent behavior. The intricate interplay between *morphology* and *behavior* must be carefully crafted together to explore how efficiently the swarm as a whole can achieve a desired collective behavior.

There is a notable body of work on concurrent design or *co-design* of morphology and behavior for individual robots [13]–[20] and most of these methods use evolutionary approach, which however suffers from computational inefficiency and consider only the bounds of morphology space without taking geometric constraints into consideration. There is limited literature on co-design in multi-robot systems [2], [21]. Most of these works are based on common simpler multi-robot problems such as foraging, aggregation,

[†] Corresponding Author, soumacho@buffalo.edu
[1] PhD Student, Mechanical and Aerospace Engineering, University at Buffalo, Buffalo, NY
[2]Associate Professor, Mechanical and Aerospace Engineering, Co-Director, Center for Embodied Autonomy and Robotics, University at Buffalo

*This work was supported by the NSF award CMMI 2048020 and the ONR grant N00014-21-1-2530. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of NSF and/or ONR.

and formation. there is also a lack of computational frameworks for co-design that allow better understanding of how swarm systems compare with single-robot systems in terms of performance and how that relates to difference in morphology or behavior. To address these gaps, this paper proposes a computational framework that enables co-optimization of morphology and behavior of individual robots in a swarm or MRS to maximize collective performance, while also allowing compare/contrast analysis of single vs. swarm for a given problem. Here, we utilize our previously proposed concept of artificial-life-inspired talent metrics [22], [23] that are physical quantities of interest, reflective of the capabilities of an individual robotic system. Talent metrics represent a compact yet physically interpretable parametric space that connects the behavior space and morphology space. We use this to decompose the morphology-behavior co-optimization into a sequence of talent-behavior optimization problems that can effectively reduce the overall search space (for each individual problem) with marginal compromise in the ability to find optimal solutions. In other words, the decomposition approach presented here is nearly lossless, i.e., a solution that can be found otherwise with a brute-force nested optimization approach to co-design will also exist in the overall search space spanned by our decomposed co-design approach (albeit assuming that each search process is ideal). We also propose a novel talent-infused policy gradient method to concurrently optimize the talents and learn the behavior.

To study operationally relevant behavior in this context, here we use a decentralized Multi-Robot Task Allocation (MRTA) problem, which finds applications in a wide range of real-world scenarios, some of which are search and rescue, disaster response, last-mile delivery, space exploration, and precision agriculture [24]–[26]. In this paper, we consider a flood response scenario in which a group of UAVs collectively supply emergency packages throughout the environment. In our previous work, we proposed a graph capsule network-based RL policy for sequential task selection in such MRTA problems [27] which demonstrated superior performance compared to other baseline methods and proved to be scalable in terms of task space [28]. Therefore, it is adopted here to guide the behavior of the multi-robot system, which will now be co-optimized alongside the morphology of the individual robots, specifically UAVs in this scenario.

Thus, the primary contributions of this paper are as follows: **1)** Present a new formulation and decomposed solution approach to concurrent (optimal) design of the morphology and learning-based behavior of multi-robot systems that are significantly more efficient than a nested co-design approach. **2)** Develop an extension of the policy architecture used to embody the behavior (decisions) of robots in MRTA to also include (morphology-dependent) talents that can be simultaneously optimized through a policy gradient process. **3)** Implement this new co-design approach to a flood response-inspired MRTA problem to identify and analyze the distinct morphology/behavior combinations obtained when using a single robot vs. using a multi-robot team (comprised of relatively simple individual robots) . In section II, we present the co-design problem formulation, and section III presents the learning-based MRTA planning approach that encompasses the behavior of the robots. Subsequently, the case study and its results are presented in Section IV, followed by concluding remarks in Section V.

II. Co-Design Framework

Consider a disaster response scenario in which a team of Unmanned Aerial Vehicles (UAVs) is deployed to deliver emergency relief supplies. The set of morphological variables, including physical form/geometry, component choices, and their physical properties (such as the motor and propeller sizes) can be expressed as $\mathbf{X}_\text{M}$. This vector comprises values $[X_1, X_2, \ldots X_n]$, each corresponding to a distinct morphological variable. These robots follow a policy or behavior denoted by $\mathbf{\Phi}$ (representing policy parameters) to efficiently plan their mission. The collective performance based on this behavior can be represented by f_C, e.g., expressing metrics such as the number of packages delivered. The primary objective of co-design is to maximize the collective performance f_C by simultaneously optimizing the morphological variables and the behavior while subject to geometric and other behavioral constraints. The optimization problem can thus be expressed as,

$$\begin{aligned} \text{Max:} \quad & f_\text{C}(\mathbf{X}_\text{M}, \mathbf{\Phi}) \\ \text{S. t.:} \quad & \mathbf{X}_\texttt{min} \le \mathbf{X}_M \le \mathbf{X}_\texttt{max}, \quad \mathbf{\Phi}_\texttt{min} \le \mathbf{\Phi} \le \mathbf{\Phi}_\texttt{max}, \quad g(\mathbf{X}_\text{M}) \le 0 \end{aligned} \tag{1}$$

where $g(\mathbf{X}_\text{M})$ represent purely morphology-dependent constraints (e.g., geometric conflicts and component incompatibilities), and $[\mathbf{X}_\texttt{min}, \mathbf{X}_\texttt{max}]$ and $[\mathbf{\Phi}_\texttt{min}, \mathbf{\Phi}_\texttt{max}]$ are respectively the (lower, upper) bounds on the morphological and behavioral (policy) parameters. Figure 1 depicts the four steps involved in our proposed co-design framework, which are explained in the later subsections.

A. Talent Metric Selection based on Morphological Constraints

During the mission, at each decision-making instance, i.e., after a package is delivered (one task completed), and the next location is determined, the robots have to consider the feasibility of proceeding to that location and completing the task. This includes assessing several factors such as the robot's remaining payload and its remaining range (ensuring it possesses enough battery to at least reach the location and return to depot afterward). These factors, which influence collective behavior, are bounded by the capability of individual robots such as max flight range and max payload capacity, which are in turn dependent on morphology. Such capabilities are used here as "*talent*" metrics, as given by:

$$\mathbf{Y}_\text{TL} = \left[Y_{\text{TL},1}, Y_{\text{TL},2}, \ldots, Y_{\text{TL},m}\right] = f_\text{M}(\mathbf{X}_\text{M}) \tag{2}$$

where f_M represents the function that maps candidate morphology variable vector to the set of m talent metrics. Typically, in model-based design, such talent metrics or properties are computed using computational analysis models.

Identifying talent metrics should follow four principles: 1) The talent metrics must be solely a function of morphology

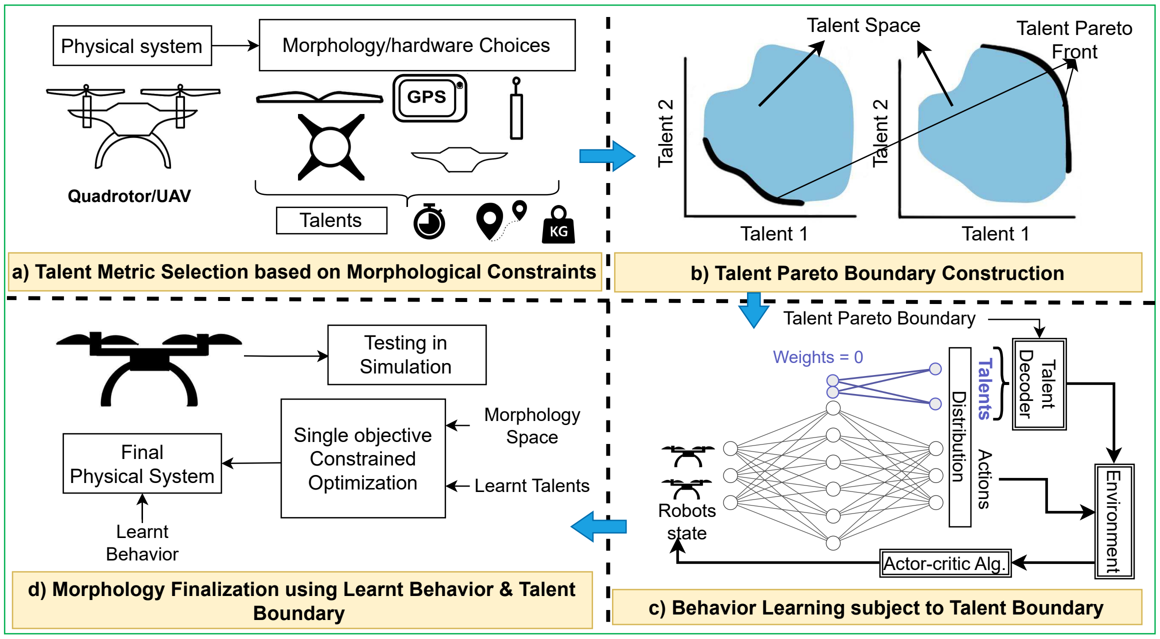

Fig. 1: Flowchart of our co-design framework; a) Morphology and its dependent talent parameters are derived; b) Based on the talents, a Pareto front is created; c) The Talent-infused policy-gradient method is used to train the associated behavior and talents; d) Final morphology is obtained via constrained optimization subject to the learnt talent and behavior.

(not affected by behavior). 2) Talent metrics should exhibit the monotonic goodness property, meaning that for each metric, there should be a consistent direction of improvement (either the greater the better, or the smaller the better). 3) Talent metrics should be collectively sufficient in computing state transitions of the system (for the given behavior context), and in determining the impact of morphology on behavior choices, meaning there cannot be a case where constraints or bounds on behavior can change with a fixed value of $\mathbf{Y}_{\text{TL}}$. 4) Talent metrics should satisfy the basic multi-objective search property, i.e., they must conflict with each other in at least part of the (morphology) design space

By adhering to these principles for identifying the talent metrics, we can reduce the computationally burdensome morphology-behavior co-optimization problem to a sequence of **1)** a multi-objective optimization to find the best trade-off talents, aka talent Pareto, and **2)** a talent/behavior co-optimization subject to not violating the determined talent Pareto. To elaborate, the second optimization must ensure that talent combinations that are beyond (or dominates) the talent Pareto front are not chosen during this process, since such combinations are in principle infeasible to achieve within the allowed morphological design space. Note that, in most robotic or complex engineering systems the dimensionality of $\mathbf{X}_{\text{M}}$ is usually much larger than that of $\mathbf{Y}_{\text{TL}}$ (the morphology space is considerably larger than the talent space), and thus this approach is also expected to enable searching a lower dimensional space during the co-optimization. This talent/behavior co-optimization process can be expressed as:

$$\begin{aligned}
\text{Max:}\quad & f_{\mathbf{C}}(\mathbf{Y}_{TL}, \mathbf{\Phi}) \\
\text{S. t.:}\quad & \min(Y_{\text{TL},1}) \le Y_{\text{TL},1} \le \max(Y_{\text{TL},1}) \\
& Q(0.05|Y_{\text{TL},1}, \cdots, Y_{\text{TL},\text{i-1}}) \le Y_{\text{TL},\text{i}} \le Q(0.95|Y_{\text{TL},1}, \cdots, Y_{\text{TL},\text{i-1}}) \\
& \forall\ i \in 2, \ldots, m-1 \\
& \mathbf{\Phi}_{\min} \le \mathbf{\Phi} \le \mathbf{\Phi}_{\max}
\end{aligned} \tag{3}$$

Here, Q represents the quantile regression model, and for every talent metric $Y_{\text{TL},\text{i}}$ except the first one (i.e., $\forall i > 1$), we progressively capture the 5th and 95th percentile values conditioned on $(Y_{\text{TL},1}, \cdots, Y_{\text{TL},\text{i-1}})$ to use it as a lower bound and upper bound of $Y_{\text{TL},\text{i}}$, respectively. For the first talent variable, we can directly acquire the bounds using the Pareto points. During co-optimization, allowed talent values must satisfy these bound constraints estimated thereof.

B. Talent Pareto Boundary Construction

Consider a set of two talent metrics. Figure 1 b) represents the feasible talent space, and based on example min-min and max-max scenarios, the lower left and upper right boundaries of this space respectively represent the Pareto front. So, for say a max-max scenario (e.g., consisting of the flight range and nominal speed of the UAV), any point further North-East of the upper right boundary is not achievable, i.e., gives an infeasible morphology candidate. In other words, the Talent Pareto not only bounds all feasible combinations of flight range and speed, but also allows us to pick best trade-off (non-dominated) combinations and ignore dominated ones – thus both constraining and reducing the search space of candidate talent combinations to consider downstream. Now, two steps are needed to identify and parametrically model this talent (Pareto) boundary:

i)*Multi-talent optimization*: A set of best trade-off talent combinations (Pareto solutions) can be obtained by solving the following multi-objective optimization problem, e.g., using a standard genetic algorithm.

$$\begin{aligned}\text{Max:}\quad & (Y_{\mathtt{TL,1}},\ldots,Y_{\mathtt{TL,m}}) = f_{\mathrm{M}}(\mathbf{X}_{\mathrm{M}})\\ \text{S. t.:}\quad & \mathbf{X}_{\mathtt{min}} \le \mathbf{X}_M \le \mathbf{X}_{\mathtt{max}},\quad g(\mathbf{X}_{\mathrm{M}}) \le 0\end{aligned} \tag{4}$$

ii) *Modeling the Pareto front*: A parametric representation of the Pareto front, f_{S}, namely the m-th talent expressed as a mathematical function of the remaining talent metrics, can be obtained by using a surrogate model such as a polynomial response surface to fit the computed talent Pareto solutions, i.e.:

$$Y_{\mathrm{TL},m} = f_{\mathrm{S}}\left(Y_{\mathrm{TL},1},\ldots,Y_{\mathrm{TL},m-1}\right) \tag{5}$$

C. Behavior Learning with Talent optimization

To generate the behavior (policy) model, we use the actor-critic method, while other standard policy gradient techniques can be exploited here as well. The structure of the policy model will depend on the nature of the behavior being learnt. A generic example of neural net based policy, aka the actor network, is shown in Fig. 1 c)(black).

***Talent-infused Actor-critic*:** To co-optimize the behavior policy along with the talents, subject to the talent Pareto boundary obtained in the previous step (Fig. 1 b)), we introduce a second small 2-layer fully connected neural net called the talent network, as shown in blue color in 1 c. The talent network architecture includes biases that are randomly initialized and pass through a linear activation layer. The resulting values are then forwarded to the output layer, which has $m-1$ neurons with sigmoid activation, where m is the number of talents. So, the talent network does not essentially have any input layers or inputs, and is defined by the biases in the first and outputs layers and weights connecting these two layers, which are the parameters optimized during training. This network is concatenated to the behavior (policy) network. Let's consider this combined network to be the new actor network. The policy of this actor network is given by $\pi((a|s,\hat{Y}_{\mathtt{TL,1}},\cdots,\hat{Y}_{\mathtt{TL,m-1}});\theta)$, where a is the behavioral action, $\hat{Y}_{\mathtt{TL,1}},\cdots,\hat{Y}_{\mathtt{TL,m-1}}$ are the talent values from the talent network and θ indicates the parameters of the combined actor network.

***Training Phase for talent-behavior co-optimization*:** In RL, a common strategy for exploration involves sampling actions from a distribution. Here, since the talents are continuous, a Gaussian distribution can be utilized. During the first step of each episode, we do a forward pass in the actor network (consisting of both the talent network and behavior network), followed by sampling from the distribution. The augmented output of the actor network is given by

$$A_\theta(s_t) = (a_t, \hat{Y}_{\mathrm{TL},1}, \hat{Y}_{\mathrm{TL,\,2}},\ldots,\hat{Y}_{\mathrm{TL,\,m\text{-}1}}),\quad \text{for all } t\in\{1,\ldots,T\} \tag{6}$$

where $A_\theta(s_t)$ signifies the output of actor policy at time step t with input state s_t, and a_t represents the action for state s_t, $\hat{Y}_{\mathtt{TL,1}},\hat{Y}_{\mathtt{TL,\ 2}},..,\hat{Y}_{\mathtt{TL,\ m-1}}$ represents the talent values from 1 to $m-1$. These $m-1$ values are subsequently processed by a talent decoder, which scales them based on the upper and lower bounds of their respective talent metric. For the first talent metric, we get:

$$Y_{\mathrm{TL},1} = \hat{Y}_{\mathrm{TL},1}(\max(\hat{Y}_{\mathrm{TL\,,\,1}}) - \min(\hat{Y}_{\mathrm{TL\,,\,1}})) + \min(\hat{Y}_{\mathrm{TL\,,\,1}}) \tag{7}$$

For remaining, 2nd to $m-1$, talents, we use the following equation:

$$\begin{aligned}Y_{\mathrm{TL},i} = \ & \hat{Y}_{\mathrm{TL},i}\,(Q(0.95|Y_{\mathrm{TL},1},\ldots,Y_{\mathrm{TL},i-1}) - Q(0.05|Y_{\mathrm{TL},1},\ldots,Y_{\mathrm{TL},i-1})) +\\ & Q(0.05|Y_{\mathrm{TL},1},\ldots,Y_{\mathrm{TL},i-1}),\quad \forall i\in\{2,\ldots,m-1\}\end{aligned} \tag{8}$$

To obtain the last talent in the set, namely, $Y_{\mathtt{TL,m}}$, we use the surrogate model created with eqn. 5 in the previous step of our co-design approach. After deciding on actions and talents, we input these into the simulation. Robots are created using these talents. Note that for the MDP computations, the robot capabilities expressed as the talent set is necessary and sufficient to model or embody the robot agent in the simulation (their morphology doesn't need to be explicitly determined). Once the talent based robot has been defined, the computed action a_t is taken to get rewards and new states, that are returned to the actor network. Crucially, after the first step of the episode, talents are not sampled from the distribution, since they are not input dependent. Moreover, changing the talent and thus the robot design during an episode would not be physically meaningful. Thus, only behavioral actions are forward propagated throughout the episode, i.e., the states and actions update with each step, as shown in the eqn 6.

The critic network, which primarily gets the states as input, is modified to receive state-talent values. Now, instead of calculating the state value, the critic network calculates the state-talent values. The new critic policy can be represented as $V(s_t,\hat{Y}_{\mathrm{TL}};w)$. The Temporal Difference (TD) error is then computed based on $\delta = r + \gamma V(s_{(t+1)},\hat{Y}_{\mathrm{TL}};w) - V(s_t,\hat{Y}_{\mathrm{TL}};w)$

Since the talent values remain the same throughout the episode, it is necessary that we collect experiences containing batches of episodes and update the actor and critic networks over this batch. The TD error can be used to update the critic to optimally estimate the state-talent value. Consequently, the actor network is updated to increase the probability of providing us with the optimal Talents and behavior (actions based on states). Once the training converges, the deterministic actor provides the optimal talents ($\mathbf{Y}^*_{\mathrm{TL}}$), and the behavior policy.

D. Morphology Finalization

Utilizing the optimized or learnt talent metrics $\mathbf{Y}^*_{\mathrm{TL}}$, we determine the final robot morphology through another single objective optimization process, as shown in fig. 1 d). The goal of this optimization is to now explicitly find the morphology that corresponds to (as closely as possible) the optimal talent metrics obtained in the previous step. This optimization can be expressed as,

$$\begin{aligned}\text{Min:}\quad & f_f = \|\mathbf{Y}_{\mathrm{TL}}(\mathbf{X}_M) - \mathbf{Y}^*_{\mathrm{TL}}\|\\ \text{S. t.:}\quad & \mathbf{X}_{\mathtt{min}} \le \mathbf{X}_M \le \mathbf{X}_{\mathtt{min}},\quad g(\mathbf{X}_{\mathrm{M}}) \le 0\end{aligned} \tag{9}$$

	Parameter
States	Task graph ($\mathcal{G}$)
	current mission time (t)
	current location of the robot (x_r^t, y_r^t)
	remaining battery of robot r (ϕ_r^t)
	capacity of robot r (c_r^t)
	destination of its peers ($x_k, y_k, k \in [1, N_R], k \neq r$)
	Remaining battery of peers ($\phi_k^t, k \in [1, N_R], k \neq r$)
	Capacity of peers ($c_k^t, k \in [1, N_R], k \neq r$)
	Destination time of peers($t_k^{next}, k \in [1, N_R], k \neq r$)
	Talents ($\hat{Y}_{\text{TL},1}$ and $\hat{Y}_{\text{TL},2}$)
Actions	Task to allocate ($0, \ldots, N_T$)

TABLE I: The state and action parameters of the graph learning policy for MRTA-Flood

		Multi-Robot					Single-Robot			
Type	Variable	Constraints		Design Outcome			Constraints		Design Outcome	
		LB	UB	Co-design	Baseline1	Baseline2	LB	UB	Co-design	Baseline
X_M	Length (m)	0.2	0.5	0.31	0.31	0.31	0.2	2.0	0.26	0.98
	Width (m)	0.2	0.5	0.31	0.31	0.50	0.2	2.0	2.0	1.98
	Motor Size (W)	100	300	100	100	300	100	1200	170	102
	Battery Size (W.h)	13.9	50	50	50	50	13.9	161	161	160
	Propeller Size (m)	0.18	0.3	0.30	0.3	0.3	0.18	1.20	0.26	0.94
	Payload(kg)	0.01	3.0	1.6	1.2	2.00	0.01	12.0	4.78	8.0
Y_n	Flight Range (km)	4.1	14.78	10.24	14.32	6.85	5.66	128	45.96	34.56
	Speed (km/hr)	19.78	41.76	22.40	19.93	25.02	12.05	90	89.99	34.40
	Package Capacity	2	7	4	3	6	1	15	12	20

TABLE II: UAV talent metrics and design variables obtained in co-design compared with baseline designs for MRTA and single robot task allocation (SRTA)

Any standard non-linear constrained optimization solver can be used here. In this paper, we use a Particle Swarm Optimization implementation [29].

III. Multi-Robot Task Allocation for Flood Response

In this work, we focus on a multi-unmanned aerial vehicle (UAV) flood disaster response problem adopted from [28], [30], which we refer to as MRTA-Flood. It consists of N_T task locations in a flood-affected area waiting for a survival package to be delivered by a team of N_R UAVs. Here, the goal is to drop survival packages to as many task locations as possible before the water level rises significantly, submerging all the locations. We assume that each location requires just one survival package. The predicted time at which a location i gets completely submerged (τ_i) is considered as the deadline of the task i, by which time that task must be completed. Each UAV has a max package (payload) capacity, max flight speed and max flight range, which comprise the set of talents. We consider a *decentralized asynchronous* decision-making scheme. The following assumptions are made: 1) All UAVs are identical and start/end at the same depot; 2) The location (x_i, y_i) of task-i and its time deadline τ_i are known to all UAVs; 3) Each UAV can share its state and its world view with other UAVs; and 4) A linear charging model with a charging time from empty to full range being 50 minutes, the charging happens every time the UAV visits the depot.

A. MRTA-Flood Problem Formulation

Here, we present a summary of the Markov Decision Process (MDP) formulation of this multi-UAV flood-response problem.

MDP over a Graph:

The MRTA-Flood problem involve a set of nodes/vertices (V) and a set of edges (E) that connect the vertices to each other, which can be represented as a complete graph $\mathcal{G} = (V, E, \Omega)$, where Ω is a weighted adjacency matrix. Each node represents a task, and each edge connects a pair of nodes. For MRTA with N_T tasks, the number of vertices and the number of edges are N_T and $N_T(N_T - 1)/2$, respectively. Node i is assigned a 3-dimensional feature vector denoting the task location and time deadline, i.e., $\rho_i = [x_i, y_i, \tau_i]$ where $i \in [1, N_T]$. Here, the weight of the edge between nodes i and j is ω_{ij} ($\in \Omega$), which can be computed as $\omega_{ij} = 1/(1 + \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (\tau_i - \tau_j)^2})$, where $i, j \in [1, N_T]$.

The MDP defined in a decentralized manner for each individual UAV (to define its task selection process) can be expressed as a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}_a, \mathcal{R} >$. The **State Space ($\mathcal{S}$)** consists of the task and peer robot properties and mission-related information. The **Action Space ($\mathcal{A}$)** is the index of the task selected to be completed next $\{0, \ldots, N_T\}$ with the index of the depot as 0. The full state and action space is shown in table I. The (**Reward function ($\mathcal{R}$)**) is defined as $10 \times N_{\text{success}}/N_T$, where N_{success} is the number of successfully completed tasks and is calculated at the end of the episode. Since here we do not consider any uncertainty, the state transition probability is 1.

B. Graph-Based Behavior Policy Network

Motivated by the generalizability and scalability benefits reported in [27], [28], [31], we construct a policy network based on specialized Graph Neural Networks (GNN) which maps the state information to an action. The behavior policy network consists of a Graph Capsule Convolutional Neural Network (GCAPCN) [32] for encoding the graph-based state information (the task graph). The remaining state information, which includes the state of the robot tasking decision, the peer robots, and the maximum range ($\hat{Y}_{\text{TL},1}$), and maximum speed ($\hat{Y}_{\text{TL},2}$), are concatenated as a single vector and passed through a linear layer to obtain a vector called the context ($F_{\text{context}} \in \mathbb{R}^{h_l \times 1}$, where h_l is the length of the context vector). The encoded and context information are then processed by a decoder to compute the actions, namely the probability of selecting each available task. Figure 2 shows the overall policy network. Further details of the GCAPCN encoder and the MHA-based decoder can be found in our previous works [27], [31], [33], and are thus not elaborated here.

IV. Case study - Results and Discussion

This section showcases the implementation and results of each step of the co-design framework applied to the MRTA-Flood problem.

A. Talent metrics and Pareto Front

For the MRTA-Flood problem, we consider a quadcopter with a Blended-Wing-Body (BWB) design integrated into

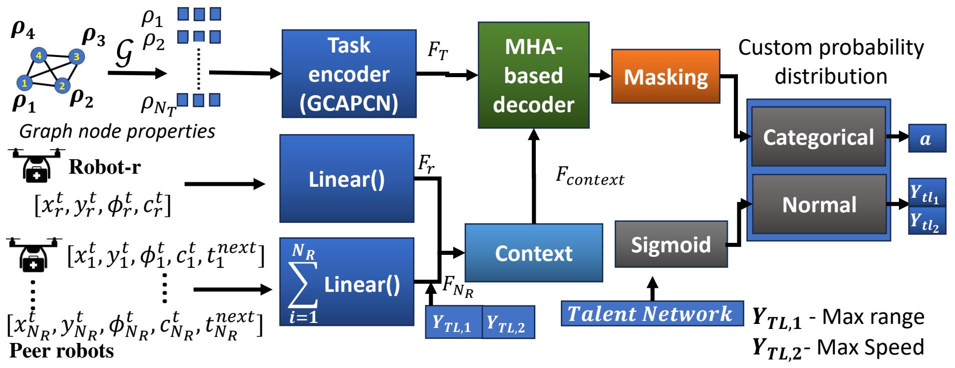

Fig. 2: The overall policy network consists of the GCAPCN encoder, context encoding, the MHA-based decoder, and Talent network.

an H-shaped frame [22]. The key morphological parameters that influence performance are the length and width of quadcopter arms, motor power, battery capacity, payload, and propeller diameters, although a much larger or more granular set of design variables can also be readily considered in future implementations. As stated earlier, the talent set comprises flight range ($Y_{\mathrm{TL},1}$), nominal speed ($Y_{\mathrm{TL},2}$), and the payload or package capacity ($Y_{\mathrm{TL},3}$) of the UAV. We consider each package to have 400 grams of emergency supplies. Computational underlying the design objective and constraint calculations (eq. 2) for this UAV can be found in [22]. In order to identify the Pareto points as explained in section II-B, we utilize the NSGA-II (Non-dominated Sorting Algorithm II) solver. For robustness, the optimization process involved conducting six separate runs, with each run consisting of a population of 120 and 40 generations each. Subsequently, the Pareto points obtained through six runs are subjected to another final non-dominated sorting process to acquire the final set of Pareto points. After the final sorting process, we identified a total of 289 Pareto solutions. Finally, to capture and model the Pareto front, we utilize 2D Quadratic regression, considering Range and Speed as independent variables and package capacity as the dependent variable, $Y_{\mathrm{TL},m}$. The resulting Pareto front is shown in figure 3 a).

B. Behavior Learning subject to Talent Boundary

We use the Stable-baselines3 [34] library to implement our custom policy and distribution as elaborated in section III. The policy generates outputs for flight range ($\hat{Y}_{\mathrm{TL},1}$) and speed ($\hat{Y}_{\mathrm{TL},2}$), both in a range of [0,1]. The flight range is scaled using the upper and lower bounds of the flight range of the Pareto front given in table II. The speed output $\hat{Y}_{\mathrm{TL},2}$ undergoes scaling based on the 5th and 95th percentile values (upper and lower limits) of speed conditioned on the flight range. The number of packages is estimated using the polynomial regression created with flight range and speed as inputs. The behavioral action (a) is also determined as a part of the policy, indicating the next task to complete. The training area is 5 sq. km, the number of robots is 5, the task size is 50, and the total mission time is 2 hours, which is kept fixed for ease of computation. For each episode, the depot, task locations, and the time deadline for each task are randomly generated across the environment.

The policy is trained using Talent-infused Proximal policy optimization (PPO) for approximately 350k episodes. Figure 4 shows the convergence history of talents and rewards. During the initial part of the training, the standard deviation of the policy is higher, and as the training progresses and rewards (Figure 4 a) start to converge, the uncertainty of talents (Figure 4 b,c,d) reduces, signifying a stable learning process. The final cumulative standard deviation of the policy narrows down to 6.9%, indicating a high level of precision and consistency in the learning outcomes.

C. Baseline Comparison

To compare our co-design policy's performance, we trained two baseline policies, and each has fixed talents: one possessing a higher package capacity and increased speed and the second baseline with a lower package capacity and higher range compared to our co-designed talents. The baseline talents are also selected from the Pareto front obtained through optimization (so they are competitive best trade-offs). The baselines represent typical automated sequential designs. Furthermore, by selecting the candidates from the Pareto front, we ensure that our co-design policy is benchmarked against design candidates that are better in one or more talents.

Identical RL settings have been used throughout the experiments in this paper. The baseline behavior policies and the co-designed policy are evaluated with 3 different task sizes and robot counts across 250 episodes each. The task completion rate by each policy is compared in Figure 6. In the training environment, which has 50 tasks and 5 UAVs, the co-designed policy demonstrated a median task completion rate of approximately 90%, outperforming the baseline policies, which achieved around 83% median task completion rate. As the environment was scaled to include 100 tasks with 10 UAVs and further to 150 tasks with 15 UAVs, the performance advantage of co-design over the baselines remained agnostic to scaling, further demonstrating the benefits of the co-design over sequential design.

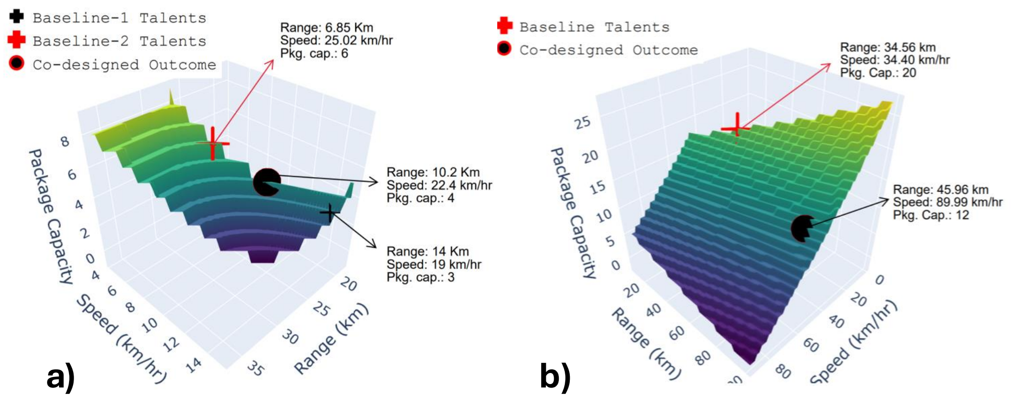

Fig. 3: Talent Pareto front approximated by polynomial regression; limits of talents captured with quantile regression. a) Pareto front for MRTA morphology constraints, b) Pareto front for SRTA morphology constraints

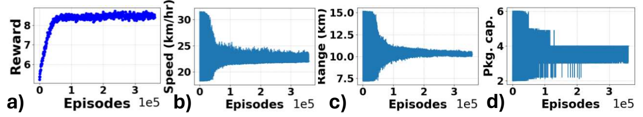

Fig. 4: Training history for MRTA co-design policy (Talents and overall reward): (a) Reward, (b) Cruise speed, (c) Flight range, (d) Package Capacity

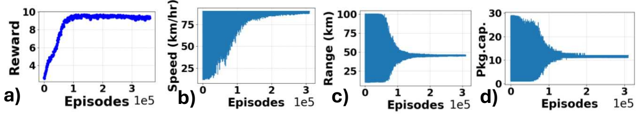

Fig. 5: Training history for Single Robot Task Allocation (Talents and overall reward): (a) Reward, (b) Cruise speed, (c) Flight range, (d) Package Capacity

D. Single Robot Task Allocation

A single robot task allocation (SRTA) co-design case study is also performed here, to provide insights with regards to: 1) At what scale of problems do co-designed multi-robot teams – by virtue of task parallelism and emergent collective performance – start providing benefits over a co-designed single robot deployment, where the latter is allowed a much larger or generous range of morphological choices (considering similar overall investment). 2) How the behavior/morphology combinations and inherent talents obtained from a single robot co-design differ from that of multi-robot co-design for the same operation.

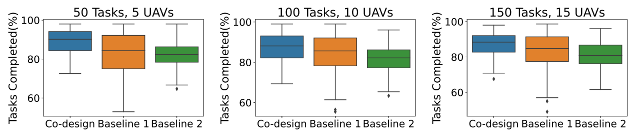

Fig. 6: Multi-Robot Case: Task completion rate of co-designed policy and baseline policies with various task and UAV Scale

E. Talent Boundaries and Behavior Learning

The upper bounds of our morphology variables are scaled 3-4 times in the single robot co-design baseline. Table II shows the upper and lower bounds of our morphology space for this single robot study. We obtained our Talent Pareto following the same method as before, and the resulting Pareto front is shown in Fig. 3 b). When compared with the Pareto front obtained for multi-robot morphology settings, the single robot Pareto front differs significantly, indicating that the influence of morphology on capability space is non-linear. The convergence history of talents and the rewards for Talent-infused learning in the single robot case are shown in Fig. 5. The co-design policy converges to a higher speed rather than a higher payload or flight range. A single UAV needs the speed to go to multiple locations and complete the tasks, while an appropriate balance between the number of packages it can carry and range is also necessary. A fixed design baseline policy is trained using talents from the single UAV Pareto front that have a higher payload than the single UAV co-designed talents. Both the baseline and optimized talents are shown in table II. In testing settings similar to the MRTA-Flood Problem, the single-robot co-designed policy surpasses the multi-robot policy in the training environment. However, its performance drops to 65% when the number of tasks double and falls below 50% when the number of tasks triple. Since scalability is an essential component in any task allocation problem, when the number of tasks is changed, multi-robot systems provide a clear advantage. This hypothesized benefit remains evident even under co-designed outcomes (which arguably bring out the near-best of both worlds, single vs. multi-robot systems).

F. Final Morphology

The final morphologies for both the single-robot task allocation and multi-robot task allocation problems are provided in table II. While the upper bounds in morphology for a single robot system are scaled 3 to 4 fold for each variable, the optimized talents do not utilize the full bounds for most parameters. Interestingly, certain variables, such as

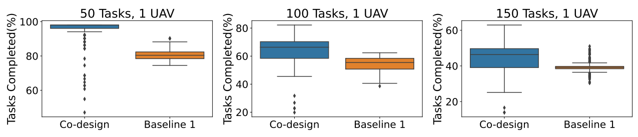

Fig. 7: Single Robot Case: Task completion rate of co-designed policy and baseline policy with various task counts

the length and propeller size, were optimized to dimensions even smaller than the morphology observed in multi-robot configurations. In order to perform a more direct comparison of single-sophisticated and multi-(simple)-robot performance and how the behavior/morphology combinations offer distinct, not necessarily intuitive, trade-offs, an anchor is needed to equate the overall investment across these cases, e.g., total cost or mass (pertinent in space applications), and would be investigated in future work.

G. Computing Costs Analysis

Our talent-behavior learning was performed in a workstation with Intel CPU-12900k (24 Threads), NVIDIA 3080ti, and 64 GB of RAM. The computation times for each step in our co-design framework for MRTA-Flood problem are: 6.7 minutes for 6 runs of NSGA-II to obtain talent Pareto solutions, just 3.5 seconds for generating the Pareto boundary regression model, 9 hours 57 minutes to train the talent-infused Actor-Critic policy, and 2.3 minutes for morphology finalization with MDPSO [29]. Overall, our co-design framework incurs a total computational cost of approximately 10 hours and 5 minutes. Using the policy training time (of 6 hours 49 minutes) with fixed morphology (namely the inner loop search) as reference, a nested co-design is estimated to take 272 hours if using NSGA-II for solving the outer level optimization.

V. Conclusion

In this paper, we introduced a new computational framework to concurrently design the learning-based behavior and morphology of individual robots in a multi-robot system, applied to the multi-robot task allocation context. Regression-based representation of the relation between the best trade-off talent choices (that represent robot capabilities), and formulation of a talent-infused actor critic policy, play key roles in enabling this new framework, with significant gains in computing efficient compared to a vanilla nested co-design approach. Applied to a multi-UAV flood response scenario, with the individual UAV behavior expressed by a graph neural network, the co-designed UAV team readily outperforms two sequential design baselines in terms task completion performance evaluated over unseen test scenarios. The framework also provides transparent insights into when a multi-UAV team becomes more beneficial compared to using a stand-alone more capable single UAV, and what morphological trades-offs occur between these two options. In its current form, the talent metrics must be purely functions of morphology, as well as be collectively sufficient to simulate the state transition underlying the robot behavior, which might be challenging to apply in settings with more complex robot/environment interactions. Future work can thus investigate talent representations that alleviate these assumptions, and thus allow wider application of the proposed co-design concept.

References

[1] L. Chen and S.-L. Ng, “Securing emergent behaviour in swarm robotics,” *Journal of Information Security and Applications*, vol. 64, p. 103047, 2022.

[2] M. Salman, A. Ligot, and M. Birattari, “Concurrent design of control software and configuration of hardware for robot swarms under economic constraints,” *PeerJ Computer Science*, vol. 5, p. e221, 2019.

[3] M. Birattari, A. Ligot, D. Bozhinoski, M. Brambilla, G. Francesca, L. Garattoni, D. Garzón Ramos, K. Hasselmann, M. Kegeleirs, J. Kuckling *et al.*, “Automatic off-line design of robot swarms: a manifesto,” *Frontiers in Robotics and AI*, vol. 6, p. 59, 2019.

[4] F. J. Mendiburu, D. G. Ramos, M. R. Morais, A. M. Lima, and M. Birattari, “Automode-mate: Automatic off-line design of spatially-organizing behaviors for robot swarms,” *Swarm and Evolutionary Computation*, vol. 74, p. 101118, 2022.

[5] A. Ligot, A. Cotorruelo, E. Garone, and M. Birattari, “Toward an empirical practice in offline fully automatic design of robot swarms,” *IEEE transactions on evolutionary computation*, vol. 26, no. 6, pp. 1236–1245, 2022.

[6] J. Kuckling, K. Ubeda Arriaza, and M. Birattari, “Automode-icepop: automatic modular design of control software for robot swarms using simulated annealing,” in *Artificial Intelligence and Machine Learning: 31st Benelux AI Conference, BNAIC 2019, and 28th Belgian-Dutch Machine Learning Conference, BENELEARN 2019, Brussels, Belgium, November 6-8, 2019, Revised Selected Papers 28*. Springer, 2020, pp. 3–17.

[7] K. Hasselmann, F. Robert, and M. Birattari, “Automatic design of communication-based behaviors for robot swarms,” in *Swarm Intelligence: 11th International Conference, ANTS 2018, Rome, Italy, October 29–31, 2018, Proceedings 11*. Springer, 2018, pp. 16–29.

[8] G. Francesca, M. Brambilla, A. Brutschy, L. Garattoni, R. Miletitch, G. Podevijn, A. Reina, T. Soleymani, M. Salvaro, C. Pinciroli *et al.*, “An experiment in automatic design of robot swarms: Automode-vanilla, evostick, and human experts,” in *Swarm Intelligence: 9th International Conference, ANTS 2014, Brussels, Belgium, September 10-12, 2014. Proceedings 9*. Springer, 2014, pp. 25–37.

[9] V. Trianni, *Evolutionary swarm robotics: evolving self-organising behaviours in groups of autonomous robots*. Springer Science & Business Media, 2008, vol. 108.

[10] H. Lipson, “Evolutionary robotics and open-ended design automation,” in *Biomimetics*. CRC Press, 2005, pp. 147–174.

[11] A. L. Christensen and M. Dorigo, “Evolving an integrated phototaxis and hole-avoidance behavior for a swarm-bot,” in *Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems. Cambridge: MIT Press. A Bradford Book*, 2006, pp. 248–254.

[12] K. Hasselmann, A. Ligot, J. Ruddick, and M. Birattari, “Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms,” *Nature communications*, vol. 12, no. 1, p. 4345, 2021.

[13] K. Sims, “Evolving 3d morphology and behavior by competition,” *Artificial life*, vol. 1, no. 4, pp. 353–372, 1994.

[14] B. Weel, E. Crosato, J. Heinerman, E. Haasdijk, and A. Eiben, “A robotic ecosystem with evolvable minds and bodies,” in *Evolvable Systems (ICES), 2014 IEEE International Conference on*. IEEE, 2014, pp. 165–172.

[15] M. Khazanov, B. Humphreys, W. D. Keat, and J. Rieffel, “Exploiting dynamical complexity in a physical tensegrity robot to achieve locomotion.” in *ECAL*. Citeseer, 2013, pp. 965–972.

[16] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, “Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding,” in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 167–174.

[17] M. Komosinski and J. Polak, “Evolving free-form stick ski jumpers and their neural control systems,” in *Proceedings of the National Conference on Evolutionary Computation and Global Optimization, Poland*, 2009, pp. 103–110.

[18] J. Bongard, “Morphological change in machines accelerates the evolution of robust behavior,” *Proceedings of the National Academy of Sciences*, vol. 108, no. 4, pp. 1234–1239, 2011.

[19] A. Gupta, S. Savarese, S. Ganguli, and L. Fei-Fei, “Embodied intelligence via learning and evolution,” *Nature communications*, vol. 12, no. 1, p. 5721, 2021.

[20] C. B. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, “Jointly learning to construct and control agents using deep reinforcement learning,” *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9798–9805, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:3352260

[21] J. Watson and G. Nitschke, “Deriving minimal sensory configurations for evolved cooperative robot teams,” in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 3065–3071.

[22] C. Zeng, P. KrisshnaKumar, J. Witter, and S. Chowdhury, “Efficient concurrent design of the morphology of unmanned aerial systems and their collective-search behavior,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 388–393.

[23] P. KrisshnaKumar, S. Paul, H. Manjunatha, M. Corra, E. Esfahani, and S. Chowdhury, “Towards physically talented aerial robots with tactically smart swarm behavior thereof: An efficient co-design approach,” *arXiv preprint arXiv:2406.16612*, 2024.

[24] P. Ghassemi, D. DePauw, and S. Chowdhury, “Decentralized dynamic task allocation in swarm robotic systems for disaster response: Extended abstract,” in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2019, pp. 83–85.

[25] C. Ju, J. Kim, J. Seol, and H. I. Son, “A review on multirobot systems in agriculture,” *Computers and Electronics in Agriculture*, vol. 202, p. 107336, 2022.

[26] L. Yliniemi, A. K. Agogino, and K. Tumer, “Multirobot coordination for space exploration,” *AI Magazine*, vol. 35, no. 4, pp. 61–74, 2014.

[27] S. Paul, P. Ghassemi, and S. Chowdhury, “Learning scalable policies over graphs for multi-robot task allocation using capsule attention networks,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8815–8822.

[28] S. Paul and S. Chowdhury, “Learning to Allocate Time-Bound and Dynamic Tasks to Multiple Robots Using Covariant Attention Neural Networks,” *Journal of Computing and Information Science in Engineering*, vol. 24, no. 9, p. 091005, 08 2024. [Online]. Available: https://doi.org/10.1115/1.4065883

[29] S. Chowdhury, W. Tong, A. Messac, and J. Zhang, “A mixed-discrete particle swarm optimization algorithm with explicit diversity-preservation,” *Structural and Multidisciplinary Optimization*, vol. 47, pp. 367–388, 2013.

[30] P. Ghassemi and S. Chowdhury, “Multi-robot task allocation in disaster response: Addressing dynamic tasks with deadlines and robots with range and payload constraints,” *Robotics and Autonomous Systems*, p. 103905, 2021.

[31] S. Paul, W. Li, B. Smyth, Y. Chen, Y. Gel, and S. Chowdhury, “Efficient planning of multi-robot collective transport using graph reinforcement learning with higher order topological abstraction,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5779–5785.

[32] S. Verma and Z. L. Zhang, “Graph capsule convolutional neural networks,” 2018.

[33] S. Paul and S. Chowdhury, “A scalable graph learning approach to capacitated vehicle routing problem using capsule networks and attention mechanism,” in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. Volume 3B: 48th Design Automation Conference (DAC), 08 2022, v03BT03A045. [Online]. Available: https://doi.org/10.1115/DETC2022-90123

[34] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-baselines3: Reliable reinforcement learning implementations,” *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html