Efficient Planning of Multi-Robot Collective Transport using Graph Reinforcement Learning with Higher Order Topological Abstraction

Steve Paul[1], Wenyuan Li[2], Brian Smyth[3], Yuzhou Chen[4], Yulia Gel[5], and Souma Chowdhury[6,†]

***Abstract*— Efficient multi-robot task allocation (MRTA) is fundamental to various time-sensitive applications such as disaster response, warehouse operations, and construction. This paper tackles a particular class of these problems that we call MRTA-collective transport or MRTA-CT – here tasks present varying workloads and deadlines, and robots are subject to flight range, communication range, and payload constraints. For large instances of these problems involving 100s-1000's of tasks and 10s-100s of robots, traditional non-learning solvers are often time-inefficient, and emerging learning-based policies do not scale well to larger-sized problems without costly retraining. To address this gap, we use a recently proposed encoder-decoder graph neural network involving Capsule networks and multi-head attention mechanism, and innovatively add topological descriptors (TD) as new features to improve transferability to unseen problems of similar and larger size. Persistent homology is used to derive the TD, and proximal policy optimization is used to train our TD-augmented graph neural network. The resulting policy model compares favorably to state-of-the-art non-learning baselines while being much faster. The benefit of using TD is readily evident when scaling to test problems of size larger than those used in training.**

I. Introduction & Motivation

Efficient solutions based on the use of multi-robot teams show increasing promise in a variety of applications ranging from disaster response [1] to manufacturing [2], warehouse logistics [3] and construction [4]. In most large-scale applications involving 100's to 1000's of tasks, using a centralized command center to perform task assignments is unlikely to be robust due to communication limitations, a single point of failure, and the likelihood of information overloading on one command center. This leads to the need for robots to take decisions in a decentralized yet timely (real-time) manner. In addition, key problem complexities include tasks with deadlines, and heterogeneity of tasks in terms of demand or workload (e.g., requiring one vs. multiple robots or trips) – such features are commonplace in the stated application scenarios. Moreover, we must account for other practical constraints, namely robot range, robot capacity (e.g., payload capacity), and limited communication range of each robot.

Given this context, in this paper, we focus on a class of multi-robot task allocation (MRTA) problems that we call MRTA-Collective Transport (MRTA-CT). MRTA-CT involves using a team of robots to perform tasks that are spatially distributed, and present time deadlines and different workloads that may require multiple visits by robots to complete the task. We posit that this problem scenario generalizes to a wide range of material transport applications, which are discussed later in this section. In addition, to introduce a sufficient degree of realism, we consider that decisions must be taken in a decentralized asynchronous manner by each robot, with robots having partial observability about the state of peer robots and the state of tasks as they start to get completed (due to communication range limitations). In addition, for ease of implementation, we assume that the robot team has full observability of tasks at the start of the operation, and a single depot (material source and recharging location) is used by the entire team.

Motivating examples: This paper focuses on MRTA-CT problems that arise in real-world operations such as: **I)** Disaster relief operations e.g., in a flood response scenario [1], where varying amounts of relief packages from a central depot must be time-efficiently delivered to victims stranded in a spatially distributed manner over the region; **II)** Manufacturing or construction sites, where a processed entity has to be delivered from a single source to multiple locations over a large site, based on their varying demand. For disaster relief, strict time deadlines are self-evident. In both scenarios, time constraints are crucial and tasks must be completed before specific deadlines. These deadlines are considered hard constraints, meaning that tasks are only considered completed if their entire demand is met before the deadline. The participating robots have a maximum payload capacity and range due to battery or fuel limitations. Any robot can partially fulfill the demand of each task location as long as the deadline has not passed. For example, if a location requires 10 relief kits and a robot can carry 5, multiple robots can deliver the kits, and a robot with remaining capacity can complete another task. If a robot carrying its full payload selects a task that only needs a few more relief kits, it can deliver them and select another task if it has enough range to do so. This approach can be used in various real-world applications, where products or raw materials must be

† Corresponding Author, soumacho@buffalo.edu

Authors [1],[2],[3],[6] are with the Department of Mechanical and Aerospace Engineering, University at Buffalo, Buffalo, NY, USA {stevepau, wli3535, briansmy, soumacho}@buffalo.edu

Author [4] is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA yuzhou.chen@temple.edu

Author [5] is with the Department of Mathematical Sciences, University of Texas at Dallas, Dallas, TX, USA ygl@utd.edu

This work was supported by the Office of Naval Research (ONR) award N00014-21-1-2530 and the National Science Foundation (NSF) award CMMI 2048020. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the ONR or the NSF.

delivered within specific timeframes for the timely execution of spatially distributed jobs that depend on those products.

A. Related Works:

While very few methods exist to directly tackle this particular MRTA-CT class of problems, there's a rich body of work on related methods in MRTA that could potentially be transitioned to this class. These methods, namely graph-matching methods [1], [5], mixed integer-linear programming (MILP) approaches [6], [7], and auction-based methods [8], [9]) typically aim to solve the ***combinatorial optimization (CO)*** problem underlying MRTA planning. However, these methods often do not scale well with the number of robots and/or tasks and do not readily adapt to complex problem characteristics without tedious handcrafting of the underlying heuristics.

In recent years, reinforcement learning or RL methods that use Graph Neural Networks (GNN) are being increasingly used to solve such planning problems with a CO formulation [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21]. This emergence of graph RL is partly attributed to the ability of GNNs to capture both Euclidean and non-Euclidean features along with local and global structural information of the task space. These methods are, however, limited in three key aspects: **1)** Simplified problems that often exclude common real-world factors such as resource and capacity constraints [10], [14], [13], [19]). **2)** Focused on smaller sized problems ($\leq$ 100 tasks and 10 robots) [22], [18]. **3)** Rarely provide evidence of generalizing to problem scenarios that are larger in size than those used for training. Such capability would be particularly critical since real-world MRTA problems often involve simulating episodes whose costs scale with the number of tasks and robots, making re-training efforts burdensome. For practical scenarios with large numbers of task locations, achieving good feasible solutions is typically the priority [23], especially under constrained communication scenarios [24] – which further motivates the need for learning-based policies to drive real-time planning in such applications.

To enable scalable policies that can be executed in real-time, a novel encoder-decoder-based RL approach was introduced by our earlier work [25], [26]. We now hypothesize that scalability can be further improved by utilizing task-neighborhood similarity. To this end, we use the Capsule Attention Mechanism or **CAPAM** policy network introduced in [25], [26], and particularly augment it with *Topological Descriptors* (TD) as novel additional task-space features to compute task-neighborhood similarity. The policy network is trained using a standard policy gradient RL algorithm, Proximal Policy Optimization (PPO) [27].

Topological Data Analysis (TDA) involves the extraction of higher-order shape features from an observed object such as graph-structured data. By shape here we broadly understand object properties that are invariant under continuous transformations such as bending, twisting, and compressing. It relies on the intuition that the extracted shape characteristics contain some inherent hidden information on the underlying object that enhance learning capabilities. The primary TD used here is Persistence Diagrams (PD) [28]. The new proposed policy network is then trained to learn sequential actions for MRTA-CT (meaning task selection by each robot taking decisions) from an output probability distribution with the overall objective to maximize the number of tasks completed. For taxonomy purposes, the MRTA-CT problem can be classified as Single-task Robots, Multi-robot Tasks, Time-extended Assignment (ST-MR-TA) class defined in [29], [30], which is an $\mathcal{NP}$-hard problem. Based on iTax taxonomy as defined in [31], this problem also falls into the In-schedule Dependencies (ID) category. The optimization formulation of the MRTA-CT as defined in this paper yields a large mixed-integer non-linear programming (MINLP) problem, which further elucidates the substantial problem complexity.

Key Contributions: The main contributions of this paper can be summarized as **1)** Formulating the MRTA-CT problems as a Markov Decision Process or MDP over graphs such that the task allocation policy can be learned using a policy gradient RL approach, where the task information is represented as node embeddings from a GNN-based encoder of the policy network, the robot states embedded as the *context* portion of the policy network, and the above two information is used to select the next task using the Attention-based decoder in a sequential manner. **2)** Explore the advantage of using higher-order structural information (encoded by TD) as additional features in improving the generalizability and scalability of the GNN-based policies. **3)** Demonstrate this learning framework's ability to generalize to larger-sized problems without the need to retrain.

The next section briefly overviews the MRTA-CT problem and its MDP formulation. Section III describes our proposed new graph learning architecture that operates on this MDP. Section IV presents the settings and outcomes of numerical experiments performed on MRTA-CT problems of varying size, used for comparative evaluation of the learning and non-learning methods, and MINLP (optimal) solutions. Section V summarizes concluding remarks and potential future extensions of our work.

II. MRTA - Collective Transport

A. Problem description

Given a homogeneous set of M robots, R=$\{r_1, r_2, \ldots, r_M\}$ and a set of N tasks V, the goal is to allocate tasks to robots for maximizing a given objective function. The objective here is to maximize the number of tasks done. There is a single depot that serves as the start and end points of each robot. Each task $i \in V$ has a unique location represented by its x-y coordinates (x_i, y_i), a workload/demand w_i (time-varying) which can be fulfilled partially by a robot, and a time deadline τ_i by which the task must be completed (demand satisfied) to consider the task i as `done` (namely ρ_i=1). Fig 1 illustrates the MRTA-CT problem. Each robot has a maximum distance range, $\Delta_{\max}$, that it can travel before returning to the depot to recharge; each robot also has a defined maximum capacity $C_{\max}$. Each robot starts from the depot where it gets a full battery and full load and then visits the task location to satisfy (fully or

partially) its demand. A robot returns to the depot once it is fully unloaded, it is running out of battery, or there are no more remaining tasks in the environment, whichever comes first. The recharging process is assumed to be instantaneous (e.g., via battery swap).

B. MRTA-CT as Optimization Problem

The exact solution to the MRTA-CT problem, excluding the communication constraints, can be obtained by formulating it as an MINLP problem, which can be concisely expressed as (for brevity):

$$\min \ f_{\text{cost}} = (N - N_{\text{success}})/N \tag{1}$$

$$N_{\text{success}} = \sum_{i \in V} \rho_i \begin{cases} \rho_i = 1, \ if \ \tau_i^f \leq \tau_i \\ \rho_i = 0, \ if \ \tau_i^f > \tau_i \end{cases}$$

$$0 \leq \Delta_r^t \leq \Delta_{\max}, r \in R \tag{2}$$

$$0 \leq c_r^t \leq C_{\max}, r \in R \tag{3}$$

Here τ_i^f is the time at which task i is completed, Δ_r^t is the available range for robot r at a time instant t, c_r^t is the capacity of robot r at time t, N_{success} is the number of successfully completed tasks during the operation. Here, we craft the objective function (Eq. (1)) such that it emphasizes maximizing the completion rate (i.e., the number of completed tasks divided by the total number of tasks). Equations 2 and 3 correspond to the remaining range and capacity respectively at time t. To learn policies that yield solutions to this CO problem, we express the MRTA-CT as an MDP over a graph, as described next.

C. MDP over a Graph

The task space of an MRTA-CT problem can be represented as a graph, including a set of nodes/vertices (V) and a set of edges (E) that connect the vertices to each other. The complete graph is given by $\mathcal{G} = (V, E, \Omega)$, where Ω is a weighted adjacency matrix. Each node represents a task, and each edge connects a pair of nodes. For MRTA-CT with N tasks, the vertices and edges are N and $N(N-1)/2$, respectively. Node i is assigned a 4-dimensional feature vector denoting the task location coordinates, time deadline, and the remaining workload/demand i.e., δ_i=$[x_i, y_i, \tau_i, w_i^t]$ where $i \in [1, N]$. Here, the weight between two edges $\Omega^{i,j}$ $(\in \Omega)$ can be computed as $\Omega^{i,j} = 1/(1 + \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (\tau_i - \tau_j)^2 + (w_i^t - w_j^t)^2})$, where $i, j \in [1, N]$.

The MDP defined in a decentralized manner for each individual robot (to capture its task selection process) can be expressed as a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}_a, \mathcal{R} >$. The components of the MDP can be defined as **State Space ($\mathcal{S}$)**, i.e., a robot r at its decision-making instance uses a state $s \in \mathcal{S}$, which contains the following information: 1) Task graph $\mathcal{G}$, 2) the current operation time t, 3) current destination of robot (x_r^t, y_r^t), 4) remaining range (battery state) of robot r δ_r^t, 5) capacity of robot r c_r^t , 6) destination of its peers $(x_k, y_k, k \in R \ k \neq r)$, 7) the remaining range of peers $\delta_k^t, k \in R \ k \neq r$, 8) capacity of peers $c_k^t, k \in R \ k \neq r$, and 9) next decision time of peers $t_k^{\text{next}}, k \in R \ \ \neq r$, 10) the time at which each peer robot took it's previous decision $ts_k, k \in R \ k \neq r$. When a robot k visits a task i the demand fulfilled by the robot k is $\min(w_i^t, c_k^t)$. **Action Space ($\mathcal{A}$):** The set of actions is represented as $\mathcal{A}$, where each action a is defined as the index of the selected task, $\{0, \ldots, N\}$ with the index of the depot as 0. The task 0 (the depot) can be selected by multiple robots, but the other tasks can be chosen if they are active (not completed or missed tasks). $\mathcal{P}_a(s'|s, a)$**:** A robot taking action a at state s reaches the next state s' in a deterministic manner. **Reward ($\mathcal{R}$):** The reward function is defined as $-f_{\text{cost}}$, calculated when there are no more active tasks (all tasks demand has been met). **Transition:** The transition is an event-based trigger. An event is when a robot reaches its selected task or visits the depot location. Here we do not consider any uncertainty, hence the state transition probability is 1.

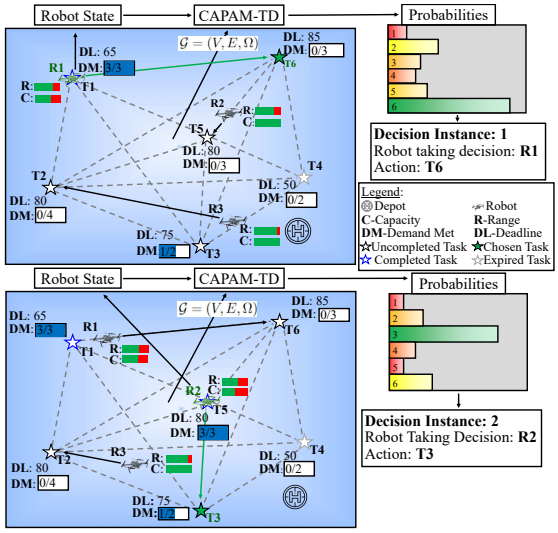


Fig. 1: Two sequential decision-making instants in MRTA-CT problem with 6 tasks and 3 robots. The task with the largest probability is chosen.

Communication modeling: The state variables 6, 7, 8, 9, and 10 are the values that robot r has about its peers during the decision-making time t, along with a vector ($w_r \in \mathbb{R}^N$) which tracks the task completion ratio, and gets updated only during an information exchange (about $(6 \times M + N) \times 8$ bytes exchanged for a `double` data type) based on the latest value of $ts_k, k \in R, k \neq r$. In our study, two robots can only communicate if their separation distance is less than a threshold distance denoted as $d_{\text{com}}^{\text{thresh}}$.

III. Learning Framework

By formulating the MRTA-CT as an MDP, we can use an RL algorithm to learn policies that maximize the objective function (Eq. 1). The learning framework mainly consists of the CAPAM-TD policy network and a policy gradient RL algorithm. The RL algorithm used here is PPO. The CAPAM policy network from [25] has demonstrated both generalizability and scalability capabilities and hence we adopt this network and enhance it with TD, with the aim to use higher dimensional topological features for decision-making to maximize the reward. In this section, we will describe the importance of the local higher-order topological information, TD using Persistent Homology, a quick overview of the CAPAM architecture, and how TD is incorporated into the

CAPAM network. The CAPAM policy network consists of a Graph Capsule Convolutional Network [32] based encoder, a context (that reads in robot states), and a Multi-Head Attention (MHA) based decoder [10]. The encoder takes in the task graph $\mathcal{G}$, computes a feature vector F_{0i} for each graph node $i \in V$ by a linear transformation of the node properties δ_i, which is then passed through multiple Graph Capsule Layers to compute permutation invariant node embeddings.

$$f_p^{(l)}(X,\mathcal{L}) = \sigma(\sum_{k=0}^{K} \mathcal{L}^k (F_{(l-1)}(X,\mathcal{L})^{\circ p}) W_{pk}^{(l)}) \quad (4)$$

Here $\mathcal{L}$ is the graph Laplacian, p is the order of the statistical moment, K is the degree of the convolutional filter, $F_{(l-1)}(X,\mathcal{L})$ is the output from $(l-1)$-th layer, $F_{(l-1)}(X,\mathcal{L})^{\circ p}$ represents p times element-wise multiplication of $F_{(l-1)}(X,\mathcal{L})$. Here, $F_{(l-1)}(X,\mathcal{L}) \in \mathbb{R}^{N \times h_{l-1} p}$, $W_{pk}^{(l)} \in \mathbb{R}^{h_{l-1} p \times h_l}$. The variable $f_p^{(l)}(X,\mathcal{L}) \in \mathbb{R}^{N \times h_l}$ is a matrix where each row is an intermediate feature vector for each node $i \in [1,N]$, infusing nodal information from $L_e \times K$ hop neighbors, for a value of p. The output of layer l is obtained by concatenating all $f_p^{(l)}(X,\mathcal{L})$, as given by:

$$F_l(X,\mathcal{L}) = [f_1^{(l)}(X,\mathcal{L}), f_2^{(l)}(X,\mathcal{L}), ...f_P^{(l)}(X,\mathcal{L})] \quad (5)$$

Here P denotes the highest order of statistical moment, and h_l denotes the node embedding length of layer l. We consider all the values of h_l (where $l \in [0, L_e]$) to be the same for this paper. Equations 4 and 5 were computed for L_e layers, where each layer uses the output from the previous layer $(F_{l-1}(X,\mathcal{L}))$. The context reads in the state of the robot taking the decision as well as the state information available to it regarding the peer robots. and computes a vector Q during a decision-making instance. The node embeddings from the encoder $F_{L_e}(X,\mathcal{L})$ and the context Q are then passed to the MHA-based decoder, which computes output probabilities for all the nodes. This output probability distribution is used to choose the next task node/task to visit by the robot taking the decision. A complete description of the CAPAM architecture can be found in our previous work [25].

A. Topological data analysis using persistent homology

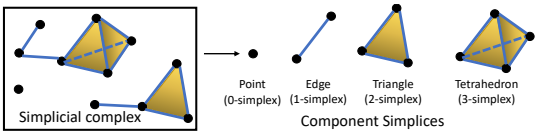


Fig. 2: A simplicial complex composed of a point, edges, triangles, and tetrahedron.

Over the past few years, there have been numerous studies demonstrating the advantage of complementing ML with topological information extracted using the machinery of TDA [33], [34], [35], where the underlying structure of data (such as a point cloud or graph) can be used for improving the learning performance. In this work, we use *Persistent Homology* (PH) [36] to extract higher-order topological information from the task graph, in the form of *Persistence Diagram* (PD). We provide here a very brief description of PH and PD. A **simplicial complex** is an entity composed of a set of points, line segments, triangles, tetrahedron, and their higher dimensional counterparts. The components of a simplicial complex are called simplicies.

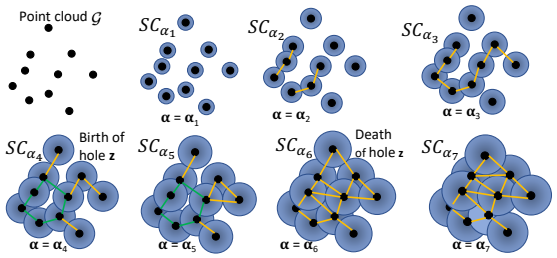


Fig. 3: PH applied to a point cloud $\mathcal{G}$. A p-dimensional hole z is characterized by α values during it's birth and death (α_4, α_6). The set which computes the birth and death α for all p-dimensional gives PD $(\mathcal{G})$

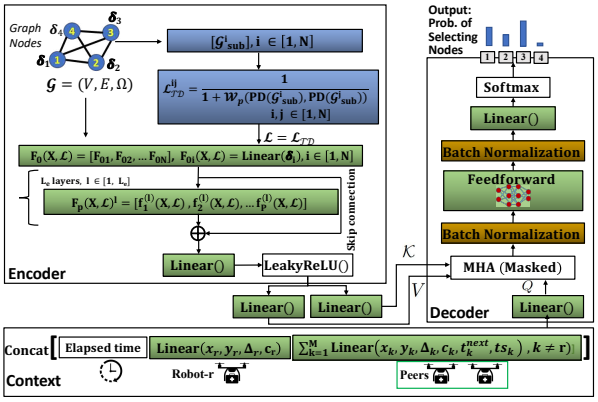


Fig. 4: CAPAM-TD policy network. Blocks in green color have learnable weights. Blocks in blue represent the TD.

For example, given a graph $\mathcal{G} = (V, E, \Omega)$, PH starts with filtration of $\mathcal{G}$ to get a new graph $\mathcal{G}_\alpha = (V_\alpha, E_\alpha, \Omega_\alpha)$, where $V_\alpha = V$, $(i,j) \in E$ if $\Omega_\alpha^{i,j} = 1$, and $\Omega_\alpha^{i,j} = 1$ if $|\delta_i - \delta_j| \leq \alpha$ else $\Omega_\alpha^{i,j} = 0$, and a simplicial complex $\mathcal{SC}_\alpha$ can be generated for $\mathcal{G}_\alpha$ (Fig. 3). Then we identify the occurrence of topological features such as cycles, cavities, and higher p-dimensional holes in $\mathcal{SC}_\alpha$ and track their lifespans. That is, as the value of α increases (starting from 0), different simplicial complexes are generated which results in the birth of different p-dimensional holes and their death. Let $\mathcal{Z}$ be the set of all the p-dimensional holes encountered with varying values of α, we track the birth and death of the p-dimensional holes using their corresponding α values. The longer the lifespan of a p-dimensional hole, the likelier this topological feature contains some latent information on the underlying object structure. All extracted topological features can be summarized in a form of PD. A PD for $\mathcal{G}$ is defined as $PD(\mathcal{G}) = \{(\alpha_1^z, \alpha_2^z) \in \mathbb{R}^2, \forall z \in \mathcal{Z}, \alpha_1^z < \alpha_2^z\}$.

B. Incorporating TDA with CAPAM

Inspired from [37], to include the local higher-order topological information, we replace the graph Laplacian $\mathcal{L}$, with a Laplacian matrix computed using PH ($\mathcal{L}_{\text{TD}} \in \mathbb{R}^{N \times N}$). First, for each node $i \in V$ of graph $\mathcal{G}$, we find a sub-graph $\mathcal{G}^i_{\text{sub}}$ which consists of k-hop neighbors of node i (where $k \geq 1$). A node $j \in V$ is an immediate neighbor of node i if $|\delta_i - \delta_j| < d_{\text{thresh}}$, where $||$ represents the L_2 norm and d_{thresh} is a threshold distance. Hence for each node $i \in V$ we can get a set of points $\mathcal{G}^i_{\text{sub}}$ (or sub-graph). For each $\mathcal{G}^i_{\text{sub}}, i \in V$

we compute it's Persistence Diagram $PD(\mathcal{G}^i_{\text{sub}})$. We use the Wasserstein distance (as in [37]) ($\mathcal{W}_p(\cdot)$) (where $0 \leq p \leq \infty$) between each node's PD to compute each element of $\mathcal{L}_{\text{TD}}$, represented as:

$$\mathcal{L}^{ij}_{\text{TD}} = \frac{1}{(1 + \mathcal{W}_p(PD(\mathcal{G}^i_{\text{sub}}), PD(\mathcal{G}^j_{\text{sub}})))}, \ \forall \ i, j \in V \quad (6)$$

Hence for each pair of nodes $i, j \in V$, $\mathcal{L}^{ij}_{\text{TD}}$ will be close to 1 if they have similar topological information, and close to 0 otherwise. Figure 4 shows the overall CAPAM-TD network.

IV. Experimental Evaluation

A. Baseline methods

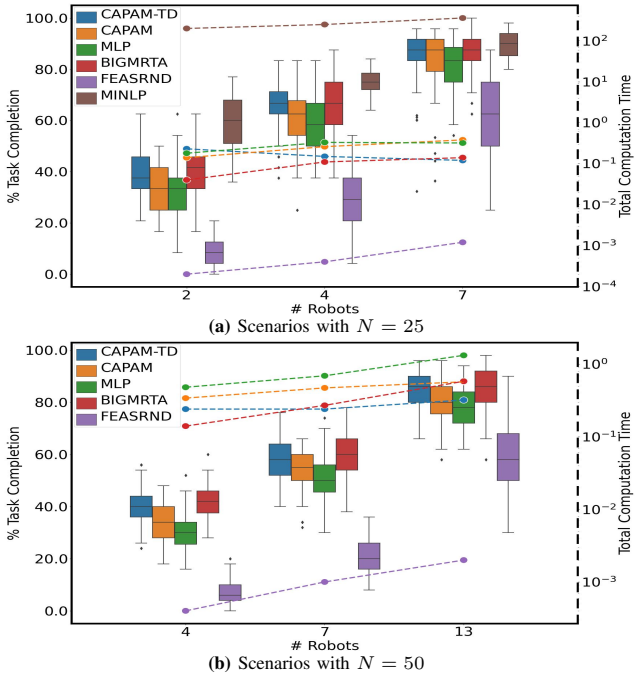
We use five baselines for performance comparison of CAPAM-TD-RL on the % task completion and total time spent for computing the decisions, with simulation time not included. These baselines include: **1) Mixed Integer Non-Linear Programming (MINLP):** We formulated the MRTA-CT as MINLP and used Gurobi [38] for solving the MINLP. Given that the MINLP solution does not consider the communication constraints, it can be used here to compute the upper bound of the optimality gap of the other methods. In other words, a successfully converged MINLP here finds ideal solutions that are as good as or better than the true optimum solutions. **2) Bi-Graph MRTA (BIGMRTA)**: BIGMRTA [39] is an online method that uses a bipartite graph to connect robots to tasks based on an incentive model. The model considers the task's features and the robot's states to determine the weights of connecting edges, which allows for the decomposition of the problem and yields a measure of robot-task pairing suitability. Each robot solves a maximum weighted matching problem to identify optimal task assignments that maximize the team's net incentive. **3) Feasibility-preserving Random-Walk (FEASRND)**: FEASRND is a myopic decision-making method that takes randomized but feasible actions, avoiding conflicts and satisfying other problem constraints. **4) Multi-Layer Perceptron based RL (MLP-RL)**: Here the node encoder of the policy network is a Multi-Layered Perceptron with 2 hidden layers (with 512 neurons), h_l =128. **5) Capsule-Attention Mechanism-based RL (CAPAM-RL)**: Here we use the CAPAM policy network from [25] with parameters as follows: K=3, L_e=3, P=3, h_l=128, (these parameters are the same for CAPAM-TD). We consider MINLP and BIGMRTA to be the upper bound performance (even though the computational time is significantly high) and FEASRND to be the lower bound.

Environment & Training details: We consider a scenario (described in section II) with a single depot, N=50, M=6 over an area of 1×1 km^2 area. Each robot has maximum payload capacity C_{max}=5kg and the demand for each task i (w_i) is drawn from a uniform distribution between 1 and 10 kg. Each robot $r \in R$ has a uniform speed of 10m/s. The time deadline for all the tasks (τ_i, $i \in V$) follows a uniform distribution between 150 and 600 seconds. The maximum range for the robots Δ_{max}=4km. The communication threshold range $d^{\text{thresh}}_{\text{com}}$=100m.

The simulation environment is developed in *Python* as an *Open AI gym* environment. The three policy networks – CAPAM-TD, CAPAM, and MLP – are trained (ensuring convergence) with the same parameters: e.g., Total steps=4×10^6, Rollout buffer=4×10^4, Batch size=4×10^3, Step size=1×10^{-6}, and Entropy coefficient = 0.01) for a fair comparison, on two GPUs (NVIDIA Tesla V100) with 16GB RAM using PPO from *Stable-Baselines3* [40]. To compute PD, we use the *Gudhi* library [41]. The trained models are tested and the non-learning (MINLP, BIGMRTA, FEASRND) based methods are implemented on a 2.6 GHz Intel core i7 MacOS 11.2.3 system.

B. Generalizability

In this paper, *generalizability* refers to the performance of the trained model on unseen test scenarios with the same (or lower) number of tasks as that of the training scenarios; and where the test and training scenarios are drawn from the same probability distribution over task locations, deadlines and demand. In this work, generalizability is evaluated on test scenarios with the number of tasks multiplied by a factor $\lambda_t \in \{0.5, 1.0\}$ and drawn from the same distribution as that of training. For each value of λ_t, we consider 3 cases with varying numbers of robots representing small, medium, and large. This gives us a total 2×3=6 scenario. For each scenario, the number of tasks N=$int(\lambda_t \times 50)$ and the number of robots M=$int(6 \times \lambda_t \times \lambda_r) + 1$, where λ_r takes a value of 0.5 for small, 1 for medium, and 2 for large number of robots. For each combination of N and M we consider 100 test samples to compare the performance, where all the 100 samples are the same for all the methods.



(a) Scenarios with $N = 25$

(b) Scenarios with $N = 50$

Fig. 5: % task completion (box plots to left axis) and average computation time in seconds (dashed lines to the right axis) for generalizability.

As expected, MINLP produced the best results for all the scenarios with N=25. However, this comes with a very high computational cost, where the samples (total 100 per scenario) were run for 200, 250, and 360 seconds for M=2,4

and 7 respectively, and without considering any communication constraints. Due to high computational expense, the MINLP solution is generated only for scenarios with N=25. For larger values of N ($\geq$ 50), the computational time per sample is very high ($>$ 1000 seconds per sample). For scenarios with N=25 and N=50 (Fig. 5), CAPAM-TD-RL outperforms CAPAM-RL, MLP-RL, and FEASRND. This is confirmed with a statistical t-test with 5% significance, p-value$<$ 0.05 (except in the case of CAPAM-RL N=25, M=7). CAPAM-RD-RL also shows a % task completion performance comparable to BIGMRTA (p-value$>$ 0.05 for all scenarios).

Even though BIGMRTA has a lower computation time for smaller-sized problems ((N=25, M=2,4), and (N=50, M=4)), for larger number of robots the computation time is higher. It is important to note the trend of the computation time. In scenarios with N=50, the computation time increases from 0.14 (M=4) to 0.58 (M=13) seconds for BIGMRTA, while for CAPAM-TD-RL the increase is from 0.24 to 0.38 seconds. This difference in the increase of computational time becomes more drastic when scaled to scenarios with larger N and M values, which is discussed in section (IV-C). CAPAM-TD-RL outperforms both learning-based methods (CAPAM-RL and MLP-RL) in terms of the average % task completion by a maximum margin of 6.5% (for N=25, M=2) and 9.9% (for N=50 and M=4) respectively. The improved performance of CAPAM-TD-RL compared to CAPAM-TD for generalizability can be credited to the use of PD to compute the graph Laplacian.

C. Scalability

Scalability refers to the performance of the trained model on test scenarios with higher numbers of tasks and robots compared to the training scenarios. Here λ_t takes values of 2, 5, and 10, while λ_r is the same as that used in the generalizability analysis. The % task completion follows a similar trend (Fig. 6) for scalability as seen in the generalizability analysis, where BIGMRTA performs the best and FEASRND performs the worst. CAPAM-TD-RL provides almost comparable performance to BIGMRTA, with the largest performance gap being 6.54 % (p-value=2e-9) for N=500, M=31, and the smallest being 0.32 % (p-value=0.86) for N=100, M=25. However, for almost all the scenarios, the computation time of CAPAM-TD-RL is significantly lower compared to BIGMRTA. For scenarios with N=500 and M=121, the average % task completion rate for BIGMRTA and CAPAM-TD-RL are 87.2% and 84.9% respectively, while their total computation times are 604 and 33 seconds, respectively. Note that BIGMRTA demonstrated comparable performance to MINLP for N=25 and λ_r=1,2, with a highest optimiality gap being just 9.4%. This shows that the ability of CAPAM-TD-RL to scale to larger problems without retraining is noticeably better than that of other learning-based methods (CAPAM-RL, MLP-RL).

V. Conclusion

In this paper, we proposed a graph RL approach to generate generalizable, scalable, and real-time executable

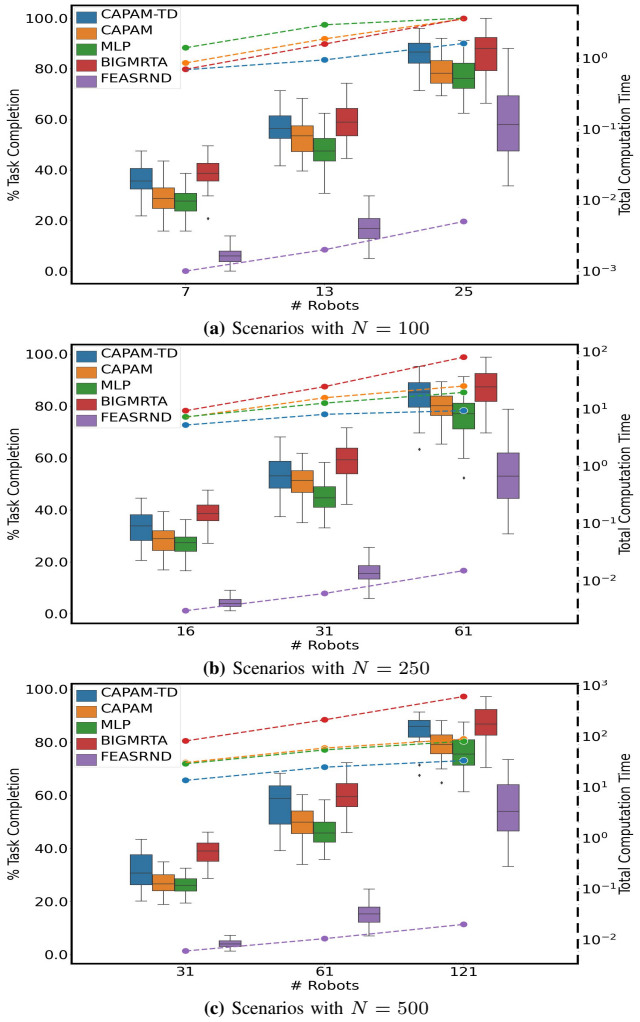


Fig. 6: % task completion (box plots to left axis) and average computation time in seconds (dashed lines to the right axis) for Scalability.

policies for a class of MRTA problems called MRTA-Collective Transport (MRTA-CT). A particular novelty of our approach lies in the introduction of Topological Descriptors (TD) as additional features that are encoded by the graph neural network serving as the policy model. Persistence homology applied to the task graph was used to derive the TD features. The proposed GNN, essentially a Capsule Attention Mechanism (CAPAM) based encoder-decoder architecture augmented with TD was trained on randomized problem samples of fixed size in terms of the number of tasks and robots, and was tested on unseen problems of varying size to assess generalizability and scalability. The performance of the CAPAM-TD model was compared to CAPAM without TD, an MLP-based RL, and 3 non-learning-based baseline methods (MINLP, BIGMRTA, and FEASRND). CAPAM-TD-RL demonstrated similar performance compared to BIGMRTA (which itself has comparable performance to MINLP for N=25) with roughly 20 times lower computation time, and better performance compared to the other baselines. The advantage of using TD is also readily evident, e.g., in the $\{N$=500, M=61$\}$ scalability test, CAPAM with TD achieves 7.1% better mean completion rate

than the one without TD.

Future directions: As the immediate next step, we intend to extend our method to problems with unreliable communication and task uncertainty, which are common features of the target applications. Further, we plan to implement the CAPAM-TD models on a more realistic multi-robot simulation environment, thereof transitioning to deployment and testing over physical testbeds.

REFERENCES

[1] 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.

[2] Y. Huang, Y. Zhang, and H. Xiao, "Multi-robot system task allocation mechanism for smart factory," in *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2019, pp. 587–591.

[3] F. Xue, H. Tang, Q. Su, and T. Li, "Task allocation of intelligent warehouse picking system based on multi-robot coalition," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 7, pp. 3566–3582, 2019.

[4] Y. Meng and J. Gan, "A distributed swarm intelligence based algorithm for a cooperative multi-robot construction task," in *2008 IEEE Swarm Intelligence Symposium*. IEEE, 2008, pp. 1–6.

[5] S. Ismail and L. Sun, "Decentralized hungarian-based approach for fast and scalable task allocation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 23–28.

[6] R. Nallusamy, K. Duraiswamy, R. Dhanalaksmi, and P. Parthiban, "Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics," *International Journal of Nonlinear Science*, vol. 8, no. 4, pp. 480–487, 2009.

[7] P. Toth and D. Vigo, *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[8] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[9] E. Schneider, E. I. Sklar, S. Parsons, and A. T. Özgelen, "Auction-based task allocation for multi-robot teams in dynamic environments," in *Conference Towards Autonomous Robotic Systems*. Springer, 2015, pp. 246–257.

[10] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[11] T. D. Barrett, W. R. Clements, J. N. Foerster, and A. I. Lvovsky, "Exploratory combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1909.04063*, 2019.

[12] S. Paul and S. Chowdhury, *A Graph-based Reinforcement Learning Framework for Urban Air Mobility Fleet Scheduling*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2022-3911

[13] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 6348–6358.

[14] Y. Kaempfer and L. Wolf, "Learning the multiple traveling salesmen problem with permutation invariant pooling networks," *ArXiv*, vol. abs/1803.09621, 2018.

[15] R. A. Jacob, S. Paul, W. Li, S. Chowdhury, Y. R. Gel, and J. Zhang, "Reconfiguring unbalanced distribution networks using reinforcement learning over graphs," in *2022 IEEE Texas Power and Energy Conference (TPEC)*, 2022, pp. 1–6.

[16] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *Advances in Neural Information Processing Systems*, 2018, pp. 539–548.

[17] A. Nowak, S. Villar, A. S. Bandeira, and J. Bruna, "A note on learning algorithms for quadratic assignment with graph neural networks," *stat*, vol. 1050, p. 22, 2017.

[18] Z. Wang and M. Gombolay, "Learning scheduling policies for multi-robot coordination with graph attention networks," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4509–4516, 2020.

[19] E. V. Tolstaya, J. Paulos, V. R. Kumar, and A. Ribeiro, "Multi-robot coverage and exploration using spatial graph neural networks," *ArXiv*, vol. abs/2011.01119, 2020.

[20] Q. Sykora, M. Ren, and R. Urtasun, "Multi-agent routing value iteration network," in *37th International Conference on Machine Learning, ICML 2020*, 2020.

[21] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, 2017.

[22] M. Strens and N. Windelinckx, "Combining planning with reinforcement learning for multi-robot task allocation," in *Adaptive Agents and Multi-Agent Systems II*, D. Kudenko, D. Kazakov, and E. Alonso, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 260–274.

[23] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Veličković, "Combinatorial Optimization and Reasoning with Graph Neural Networks," 2021.

[24] H. Cao, S. Lacroix, F. Ingrand, and R. Alami, "Complex tasks allocation for multi robot teams under communication constraints," 2010.

[25] 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)*, 2022, pp. 8815–8822.

[26] S. Paul and S. Chowdhury, "A scalable graph learning approach to capacitated vehicle routing problem using capsule networks and attention mechanism," *ASME 2022 International Design Engineering Technical Conferences (IDETC 2022)*. [Online]. Available: https://par.nsf.gov/biblio/10345362

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[28] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of persistence diagrams," in *Proceedings of the twenty-first annual symposium on Computational geometry*, 2005, pp. 263–271.

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

[30] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, pp. 55–70, 2017.

[31] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

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

[33] C. S. Pun, K. Xia, and S. X. Lee, "Persistent-homology-based machine learning and its applications–a survey," *arXiv preprint arXiv:1811.00252*, 2018.

[34] J. Townsend, C. P. Micucci, J. H. Hymel, V. Maroulas, and K. D. Vogiatzis, "Representation of molecular structures with persistent homology for machine learning applications in chemistry," *Nature communications*, vol. 11, no. 1, pp. 1–9, 2020.

[35] Z. Cang and G.-W. Wei, "Integration of element specific persistent homology and machine learning for protein-ligand binding affinity prediction," *International journal for numerical methods in biomedical engineering*, vol. 34, no. 2, p. e2914, 2018.

[36] M. E. Aktas, E. Akbas, and A. E. Fatmaoui, "Persistence homology of networks: methods and applications," *Applied Network Science*, vol. 4, no. 1, pp. 1–28, 2019.

[37] Y. Chen, B. Coskunuzer, and Y. Gel, "Topological relational learning on graphs," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 029–27 042, 2021.

[38] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: http://www.gurobi.com

[39] 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)*. New Brunswick, NJ: IEEE, aug 2019, pp. 83–85. [Online]. Available: https://ieeexplore.ieee.org/document/8901062/

[40] 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

[41] The GUDHI Project, *GUDHI User and Reference Manual*, 3.6.0 ed. GUDHI Editorial Board, 2022. [Online]. Available: https://gudhi.inria.fr/doc/3.6.0/

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[43] T. Brodeur, P. Regis, D. Feil-Seifer, and S. Sengupta, "Search and rescue operations with mesh networked robots," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2018, pp. 6–12.

[44] S. Shelly and A. V. Babu, "A probabilistic model for communication link reliability in vehicular ad hoc networks," in *2014 IEEE International Conference on Vehicular Electronics and Safety*, 2014, pp. 123–128.

APPENDIX

A. Mixed Integer Non-Linear Programming Formulation for MRTA-Collective Transport Problem

NOMENCLATURE

(h, s, r) Tuple representing robot r during decision number h of route s

(i, j, h, s, r) Tuple representing a scenario where robot r travels from location i to j during decision number h of route s

(j, h, s) Tuple representing task j during decision number h of route s

$\Delta(h, s, r)$ Range of robot r during (h, s, r)

Δ_{max} Maximum allowed range in a single tour for each robot

τ_i Time deadline for task i

time(i, j, h, s, r) time taken to execute scenario (i, j, h, s, r)

$\text{time}_i^{\text{complete}}$ Time at which task i is completed

$c(h, s, r)$ Capacity of robot r after scenario during scenario (h, s, r)

C_{max} Maximum capacity of each robot

$d(i, j)$ Distance between nodes i and j

D Depot

$e(i, j, h, s, r)$ work done during scenario (i, j, h, s, r)

H Maximum number of decisions per tour for each robot

h Index for each decision in a tour

i, j, k Indices for nodes

M Number of robots

N Number of tasks

R Set of robots

r Index for robots

S Maximum number of tours for each robot

s Index for each tour

$t(i, j)$ Time to reach from node i to j

T Set of tasks

V Set of tasks including the depot, [D,T]

$w(j, h, s)$ demand met during scenario (j, h, s)

w_i^{act} Demand for task i

$x(i, j, h, s, r)$ Binary decision variable which takes a value of 1 if robot r travels from node i to j during decision number h of route s

1) Objective Function and Constraints: Objective function eq. 7 is set to maximize the number of tasks done, subject to a set of constraints.

$$\max \ \frac{N_{\text{success}} - N}{N} \tag{7}$$

$$\sum_{j \in V} x(1, j, 1, s, r) = 1, \ \forall s \in [1, S], \ \forall r \in R \tag{8}$$

$$\sum_{j \in V} x(j, 1, H, s, r) = 1, \ \forall s \in [1, S], \ \forall r \in R \tag{9}$$

$$\sum_{i,j \in V} x(i, j, h, s, r) \leq 1, \ \forall h \in [1, H], \ \forall s \in [1, S], \ \forall r \in R \tag{10}$$

$$\sum_{j \in V} x(i, j, h, s, r) = \sum_{k \in V} x(k, i, h-1, s, r), \quad \forall i \in V, \ \forall h \in [2, H] \ \forall s \in [1, S], \ \forall r \in R \tag{11}$$

$$\Delta(1, s, r) = \Delta_{max} - \sum_{i,j \in V} x(i, j, 1, s, r) \times d(i, j), \quad \forall s \in [1, S], \ \forall r \in R \tag{12}$$

$$0 \leq \Delta(h, s, r) \leq \Delta_{max}, \ \forall i, j \in V, \quad h \in [1, H], \ s \in [1, S], \ r \in R \tag{13}$$

$$0 \leq c(h, s, r) \leq C_{max}, \ \forall i, j \in V, \quad h \in [1, H], \ s \in [1, S], \ r \in R \tag{14}$$

$$\Delta(h, s, r) = \Delta(h-1, s, r) - \sum_{i,j \in V} x(i, j, h, s, r) \times d(i, j), \quad \forall h \in [2, H], \ \forall s \in [1, S], \ \forall r \in R \tag{15}$$

$$w(j, 1, 1) = \sum_{i \in V, \forall r \in R} e(i, j, 1, 1, r) \times x(i, j, 1, 1, r), \quad \forall j \in V \tag{16}$$

$$w(1, s, r) = w(H, s-1, r) + \sum_{\forall i \in V, \forall r \in R} e(i, j, 1, s, r) \times x(i, j, 1, s, r), \quad \forall j \in V, \ \forall s \in [2, S] \tag{17}$$

$$0 \leq e(i, j, h, s, r) \leq C_{max}, \ \forall i, j \in V, \quad \forall h \in [1, H], \forall s \in [1, S], \ \forall r \in R \tag{18}$$

$$e(i, j, h, s, r) \leq c(h-1, s, r), \ \forall i, j \in V, \quad \forall h \in [2, H], \forall s \in [1, S], \ \forall r \in R \tag{19}$$

$$e(i, j, h, s, r) \leq w_j^{act} - w(j, h, s), \ \forall i, j \in V, \quad \forall h \in [2, H], \forall s \in [1, S], \ \forall r \in R \tag{20}$$

$$w(j, h, s) = w(h-1, s, r) + \sum_{\forall i \in V, \forall r \in R} e(i, j, h, s, r) \times x(i, j, h, s, r), \ \forall j \in V, \quad \forall h \in [2, H], \forall s \in [2, S] \tag{21}$$

$$c(1,s,r) = C_{max} - \sum_{i,j \in V} e(i,j,1,s,r) \times x(i,j,1,s,r), \quad \forall s \in [1,S] \, \forall r \in R \tag{22}$$

$$c(h,s,r) = c(h-1,s,r) - \sum_{\forall i,j \in V} e(i,j,h,s,r) \times x(i,j,h,s,r), \quad \forall h \in [2,H], \forall s \in [1,S], \ \forall r \in R \tag{23}$$

$$w(j,h,s) \leq w_j^{act}, \ \forall j \in V, \quad \forall h \in [1,H], \forall s \in [1,S], \ \forall r \in R \tag{24}$$

$$\sum_{i \in V, r \in R} w(j,H,S) = w_j^{act}, \ \forall j \in V \tag{25}$$

$$\text{time}(i,j,h,s,r) = t(i,j) \times x(i,j,h,s,r), \ \forall i,j \in V, \quad \forall h \in [1,H], \forall s \in [1,S], \ \forall r \in R \tag{26}$$

$$\text{time}_j^{\text{complete}} = \sum_{i \in V, h \in [1,H], s \in [1,S], r \in R} \text{time}(i,j,h,s,r), \quad \forall i,j \in V \tag{27}$$

$$N_{\text{success}} = \sum_{i \in V-1} \text{Done}_i \begin{cases} \text{Done}_i = 1, \ if \ time_i^{\text{complete}} \leq \tau_i \\ \text{Done}_i = 0, \ if \ time_i^{\text{complete}} > \tau_i \\ \forall i \in V-1 \end{cases} \tag{28}$$

Constraints 8 and 9 ensures that every robot start and ends a route/tour from the depot. Constraint 10 ensures that each robot can make a maximum of one transition during each decision-making instances. Constraint 11 makes the start location of a transition same as the end location of the previous transition for each robot. Constraint 12 sets the range of the robots as Δ_{max} during the start of a journey. The range update after each transition for the robots are governed by equations 13 and 15. Constraints 18, 19, and 20 ensures that the work done by a robot do not exceed it's capacity the demand of it's location, while constraints 17 and 21 corresponds to the update for the demand met for each task. Constraints 24 and 25 enforces the the demand met does not exceed the actual demand. Constraints 22 and 23 corresponds to the capacity update for the robots. Constraints 26, 27, and 28 are used to compute the task completion time for each task and the number of tasks completed before deadline.

B. More details on CAPAM and PH

1) CAPAM architecture: The CAPAM policy architecture proposed in [25] consists of three parts, which are the encoder, context, and decoder.

Encoder: The encoder takes in the task information which is represented as a graph $\mathcal{G}$, and computes node embeddings for each node $i \in V$ using the Graph Capsule layers as shown in equations 4 and 5 in section III.

Context: The context consists of the following features: **1)** elapsed mission time; **2)** range of the robot taking decision; **3)** capacity of the robot taking decision; **4)** current location of the robot taking decision; **5)** current destination of robot's peers; **6)** range capacity of peer; **7)** work capacity of peer robots. These features are transformed and aggregated as single learnable vector of length h_q, which then undergoes a linear transformation to get a vector of length h_l also called the query Q. It should be noted that features 5,6,and 7 are the state of the peer robots which the robot taking the decision has during the decision-making instance.

Decoder: The MHA-based decoder use the information from the encoder and the context or query, and thereof choose the best task by calculating the probability value of getting selected for each (task) node. In this case, the first step is to feed the embedding for each node (from the encoder) as **key-values** ($\mathcal{K}$, V), since inputs for MHA are key-value pairs [42]. The key K and value V for each node is computed by two separate linear transformations of the node embedding obtained from the encoder. Now the attention mechanism can be described as mapping the query (Q) to a set of key-value ($\mathcal{K}, V$) pairs. The inputs, which are the query (Q) is a vector, while $\mathcal{K}$ and V are matrices of size $h_l \times N$ (since there are N nodes). The output is a weighted sum of the values V, with the weight vector computed using the compatibility function expressed as:

$$\text{Attention}(Q,\mathcal{K},V) = \text{softmax}(Q^T\mathcal{K}/\sqrt{h_l})V^T \tag{29}$$

where h_l is the dimension of the key of any node i ($k_i \in \mathcal{K}$). The output from each MHA layer is obtained as:

$$\text{MHA}(Q,\mathcal{K},V) = \text{Linear}(\text{Concat}(\text{head}_1 \dots \text{head}_{h_e})) \tag{30}$$

Here $\text{head}_i = \text{Attention}(Q,\mathcal{K},V)$ and h_e (taken as 8 here) is the number of heads. A final `softmax` layer outputs the probability values for all the nodes. The nodes which are already visited will be masked (by setting their probability as 0) so that these nodes are not available for selection in the future time steps of the simulation of the multi-robot operation.

2) Persistent Homology (PH): The Persistence Diagram (PD) of a point cloud $\mathcal{G} = (V, E, \Omega)$ is computed by applying PH over $\mathcal{G}$. Here we consider $\mathcal{G}$ to be an undirected graph. The next step is to apply a filtration technique to find topological features such as cycles (2-D hole), cavities (3-d hole), and higher dimensional holes. These holes features come into existence as edges are formed between the nodes of the graph. Consider a threshold distance α such that two nodes u and v has an edge between them if $|\delta_u - \delta_v| \leq \alpha$, where δ_u and δ_v represents the features of nodes u and v, respectively. As the value of α increases, new edges are formed which results in new simplical complexes. We track the birth and death of all the k-dimensional holes. For example, from figure 3, a 2-d z is born at $\alpha = \alpha_4$. This hole persists until $\alpha = \alpha_6$ where two new edges are formed and as a result hole z disappears, which is considered at it's death. Therefore (α_4, α_6) denotes the persistence of z.

Similarly this performed for all the k-dimesional holes $\mathcal{O}$. Therefore the persistence diagram of $\mathcal{G}$ is represented as $PD(\mathcal{G}) = (\alpha_o^1, \alpha_o^2) \forall o \in \mathcal{O}$, where α_o^1 and α_o^2 represents the birth and death of $o \in \mathcal{O}$.

C. Communication Modelling

In a real world scenario, a full communication between each robots in a mission is not always possible. Each robot will only be able to communicate with robots on close proximity. Robots deployed for disaster response will be implemented with a communication device such as Wifi or Zigbee [43] with a typical range of 'x' meters. The communication between two robots a and b is two way, where robot 'a' shares the information it has about the environment and other robots, to robot 'b', and vice versa. This information exchange can be modelled in a probabilistic manner similar to [44], where the information exchange does not happen beyond a threshold distance, and within the threshold distance, the information exchange happens probabilistically, where smaller the separation distance larger is the probability of information exchange. In order to simplify the communication modelling, we assume that information exchange occurs when the separation distance between two robots at a time instant is less than a threshold distance d_{com}^{thresh}, and does not occur if the separation distance is greater than d_{com}^{thresh}. In this paper, we consider $d_{com}^{thresh} = 100\ meters$. Apart from the static information such as the location of the tasks and its time deadline, each robot r keeps a record of 1) information regarding the completion of a task $visited_r$, 2) information regarding all the robots including (self information) $RobotState_r$.

Each robot $r \in R$ maintains vector ($completion_r$) of size $1 \times N$, where each element of $completion_r$ corresponds to a task i, and represents the fraction of the total demand met for task i. For example, during the information exchange between two robots a and b, robot a updates $visited_a$ as, $visited_a$ = $visited_a \mid visited_b$, where '|' represents a pairwise logical '*or*' operator.

The self state information generated by $r \in R$ includes 1) destination coordinates (x_r, and y_r), 2) current available range Δ_r , 3) current payload capacity c_r 4) a time stamp when this information was generated t. Let $RobotState_r^l$ be the state information of robot l that robot r has during an instance. So we can express $RobotState_r^r = [x_r, y_r, \Delta_r, c_r, t]$. Therefore for all robots $r \in R$, $RobotState_r$ is a matrix of size $M \times 5$, where each row l represents $RobotState_r^l$.

While information exchange occurs between two robots a and b, the state information update happens only for those entries with newer timestamps. For example, during an information exchange between robots a and b, robot a has both $RobotState_a$ and $RobotState_b$, and robot a will replace $RobotState_a^l$ ($l \in R$) with $RobotState_b^l$ if $RobotState_b^l$ has a newer timestamp compared to that of $RobotState_a^l$.