LEARNING TO SOLVE MULTI-ROBOT TASK ALLOCATION WITH A COVARIANT-ATTENTION BASED NEURAL ARCHITECTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper presents a novel graph (reinforcement) learning method to solve an important class of multi-robot task allocation (MRTA) problems that involve tasks with deadlines, and robots with ferry range and payload constraints (thus requiring multiple tours per robot). While drawing motivation from recent graph learning methods that learn to solve combinatorial optimization problems of the mTSP/VRP type, this paper seeks to provide better convergence and tractable scalability (with an increasing number of tasks) in learning to solve MRTA. The proposed neural architecture, called Covariant Attention-based Model or CAM, includes two main components: 1) an encoder: a covariant node-based embedding model to represent each task as a learnable feature vector, and 2) a decoder: an attention-based model to facilitate a sequential output. In order to learn the feature vectors, a policy-gradient method based on REINFORCE is used. The new learning architecture is applied to a flood response problem (representative of the general class of Multi-task Robots, and Single-robot Tasks or MR-ST problems), where multiple unmanned aerial vehicles (UAVs) supply survival kits to victims spread out over an area. To perform a comparative analysis, the well-known attention-based approach (that is designed to solve mTSP/VRP problems) is extended and applied to the stated MRTA class of problems. A comparison is performed in terms of a cost function that considers both task completion rate and cumulative traveled distance. The results show that our primary method is superior not only in terms of the cost function (over various training and unseen test scenarios), but also provide significantly faster convergence and yields learnt policies that can be executed within 6 milliseconds per robot, thereby allowing real-time application.

1 INTRODUCTION

In multi-robot task allocation (MRTA) problem, we study how to coordinate tasks among a team of cooperative robotic systems such that the decisions are free of conflict and optimize a quantity of interest (Gerkey & Matarić, 2004). This problem has been studied in the last decades and it is getting more attention recently due to its potential real-life applications, such as, disaster response (Ghassemi & Chowdhury, 2018), last-mile delivery (Aurambout et al., 2019), environment monitoring (Espina et al., 2011), and reconnaissance (Olson et al., 2012)). Although various approaches (e.g., graph-based methods (Ghassemi & Chowdhury, 2018; Ghassemi et al., 2019), integer-linear programming (ILP) approaches (Nallusamy et al., 2009; Toth & Vigo, 2014; Cattaruzza et al., 2016; Jose & Pratihar, 2016), and auction-based methods (Dias et al., 2006; Schneider et al., 2015)) have been proposed to solve MRTA problems, they are not easily extendable to solve non-deterministic MRTA problems. In the recent years, a rich body of work has emerged on using learning-based techniques to model solutions or intelligent heuristics for combinatorial optimization (CO) problems (MRTA is a class of CO problems) over graphs. The existing works are limited to general classical CO problems, such as multi-traveling salesmn (mTSP), vehicle routing (VRP), and max-cut type of problems. In this work, we are interested in a class of MRTA problem with additional physical constraints, including tasks with time deadline, and robots with constrained payload and ferry-range.

For this purpose, we aim to demonstrate the effectiveness of formulating a MRTA problem in graph space and learn the optimal policies using a reinforcement learning (RL) method. In this work, we are specifically interested in a class of MRTA problem that falls into the Multi-task Robots, and Single-robot Tasks (MR-ST) class defined in (Gerkey & Matarić, 2004; Nunes et al., 2017) (i.e., a feasible and conflict-free task allocation is defined as assigning any task to only one robot (Ghassemi et al., 2019)). Specifically, in our MR-ST problem, we consider the following constraints and challenges: (1) payload capacity and operating range of robots or UAVs; and (2) task completion deadlines (e.g., victim response deadlines). In this paper, we propose a new covariant attention-based model (aka CAM) for learning over graphs, by incorporating an embedding where it is permutation-invariant and it can capture multi-scale features.

1.1 Multi-Robot Task Allocation

The MRTA problem can be formulated as an Integer Linear Programming (ILP) or mixed ILP. When tasks are defined in terms of location, the MRTA problem becomes analogical to the Multi-Traveling Salesmen Problem (mTSP) (Khamis et al., 2015; Svestka & Huckfeldt, 1973) and its generalized version, the Vehicle Route Planning (VRP) problem (Dantzig & Ramser, 1959). Existing solutions to mTSP and VRP problems in the literature (Bektas, 2006; Braekers et al., 2016) have addressed analogical problem characteristics of interest to MRTA, albeit in a disparate manner; these characteristics include limited vehicle capacity, tasks with time deadlines, and multiple tours per vehicle, with applications in the operations research and logistics communities (Azi et al., 2010; Baldacci et al., 2011; Mingozzi et al., 2013; Wang et al., 2018). ILP-based mTSP-type formulations and solution methods have also been extended to task allocation problems in the multi-robotic domain (Jose & Pratihar, 2016; Kamra & Ayanian, 2015; Liu & Shell, 2013). Although the ILP-based approaches can in theory provide optimal solutions, they are characterized by exploding computational effort as the number of robots and tasks increases (Nallusamy et al., 2009; Toth & Vigo, 2014; Schneider et al., 2015; Cattaruzza et al., 2016; Jose & Pratihar, 2016). Due to their computationally expensive, they are suitable for offline or (at intermediate scales) centralized offboard deployment (Korsah et al., 2013), and not online planning. Recently, it has been shown that Graph Neural Networks (GNNs) can be used to provide an alternative method with computational efficient run-time (Kool et al., 2019).

1.2 Learning over Graphs

The ANN or deep learning based methods for learning CO can be broadly classified into: (i) Reinforcement Learning (RL) methods (Kool et al., 2019; Barrett et al., 2019; Khalil et al., 2017); and (ii) supervised learning (often combined with RL) methods (Kaempfer & Wolf, 2018; Mittal et al., 2019a; Li et al., 2018; Nowak et al., 2017). The supervised learning approaches typically address problem scenarios where samples are abundant (e.g., influence maximization in social networks (Mittal et al., 2019a)) or inexpensive to evaluate (e.g., TSP (Kaempfer & Wolf, 2018)), and are thus unlikely to be readily applicable to solve complex problems over real-world graphs. RL based techniques to learn on graphs include attention models with REINFORCE (Kool et al., 2019) and deep Q-learning (Khalil et al., 2017; Barrett et al., 2019), among others, with some extending solutions to multi-agent settings (Jiang et al., 2020) In this work, we are interested in the first class of the methods (i.e., RL methods over graph space).

Dai et al. (2017) showed that a combination of graph embedding and RL methods can be used to approximate optimal solutions for combinatorial optimization problems, as long as the training and test samples are drawn from the same distribution. Mittal et al. (2019b) presented a new framework to solve a combinatorial optimization problem. In this framework, Graph Convolutional Network (GCN) performs the graph embedding and Q-Learning learns the policy. The results demonstrated that the proposed framework is able to learn solve unseen test problems if they have been drawn from the same distribution of the training dataset. More importantly, it has been shown that using a learnt network policy instead of tree search, both methods are using the same embedding GCN, showed a speedup of 5.5 for a problem size of 20,000. Similarly, the effectiveness of learning a network policy using Q-Learning to solve the Max Cut problem (a combinatorial problem) has been demonstrated by Barrett et al. (2019).

Recently, it has been a growing interest in using sequence-to-sequence models (e.g., pointer networks and attention mechanism) to encode and learn the combinatorial optimization problems in

graph space (Kaempfer & Wolf, 2018; Kool et al., 2019). Kool et al. (2019) created a framework using an encoder/decoder based on attention mechanism and REINFORCE algorithm for solving a wide variety of combinatorial optimization problem as graphs, with the main contribution being flexibility of the approach on multiple problems with the same hyper parameters. Majority of the above methods solve classical problems without physical, real-world constraints (i.e., mTSP, VRP and Max-Cut type of problems). Taking into account additional physical characteristics such as robots with payload capacity and range constraints and tasks with time deadline requires additional steps, which motivates this research.

In this paper, a new method is proposed that uses attention mechanism with an enhanced encoding network (embedding layer), which can capture complex and multi-scale features. The embedded layer is a variation of Covariant Compositional Networks (CCN), introduced by Hy et al. (2018). This node-based embedding has been chosen since it: (i) operates on undirected graph; (ii) uses receptive field and aggregation functions based on tensor product and contraction operations (which are equivariant), this leads to a permutation- and rotation-invariant embedding; and (iii) uses tensors, which provides a multi-scale representation (n-th order tensor representation), this can be used to extend the work for multi-level networks. An exact version of the CCN method is not tractable for a large problem. Hence, a variation of this method has been proposed in this work.

1.3 Objectives of this Paper

In this paper, we consider each task for the robot to be going to a location and then dropping a survival kit payload. Unlike a single agent combinatorial optimization problem, a multi-agent combinatorial optimization has additional computational complexities. Apart from the multi-agent feature, we are also considering additional constraints for the tasks and the robots/agents. For each task i in N, it has a time deadline t_i, by which the task has to be done in-order for it to be considered as successfully done. The constraints for the robots are: 1) Each robot has a limited range of distance that it can travel after fully recharging, 2) Each robot has a maximum capacity of payload it can take. We also assume that there is a depot, where each robot starts the mission, get recharged when running out of energy, and refill the payload capacity to visit more locations to complete the remaining tasks.

The primary contributions of this paper are: 1) Extending attention-based mechanism (AM) method (Kool et al., 2019) to solve a multi-agent combintorial optimization; and 2) Proposing a new graph neural network architecture by incorporating an attention-based decoder and a covariant node-based embedding model as encoder. This new architecture should be able to capture the complexity of the multi-agent CO problem and hence improve the performance of the learnt model. The remaining portion of the paper is organized as follows: The next section presents our proposed new graph neural network for the MRTA problem. Section 4 describes simulation settings and different case studies. Results, encapsulating the performance of these methods on different-sized problems and various analyses of the proposed method, are presented in Section 5. The paper ends with concluding remarks.

2 Covariant Attention-based Neural Architecture

The MRTA problem that we consider in this paper has a fixed number of tasks N and a fixed number of robots N_r for completing the tasks. Each task has a location specified by its x and y coordinates. Here we assume that a robot goes to the location of the task and drops a package as the exact task. For simplicity we assume that if a robot reaches the location of the task before the time deadline of the task, then it is considered to be successfully done, else we consider it to be unsuccessful. Our proposed covariant attention-based model, the so-called CAM, consists of an encoder and an decoder. MRTA problem consist of multiple tasks located in different location, with each location having a x coordinate, y coordinate, and a time deadline within which the task has to be completed. Therefore its possible to consider MRTA problem as a graph with each node having the x coordinate, y coordinate, and time deadline (t), as the node properties. For learning over graphs we need to represent each node as a continuous vector, preserving its properties and its structural information. We designed an encoding algorithm explained in Section 2.2. We then use an attention-based decoder (Section 2.3) which takes in the node embeddings, and a vectorized representation of the current state (which includes the state of the mission and the state of all the robots) sequentially, to get a probability distribution of the next node to visit (for the robot that is taking a decision on which task to do next). Both the encoder and the decoder has learnable parameters which will be learned using the learning algorithm described in Section 3.

2.1 MODELLING MRTA PROBLEM AS GRAPH

Let N be the number of tasks in a MRTA problem. The N tasks in the MRTA can be represented as a graph with N nodes.

$$\phi = G(V, E) \tag{1}$$

where V, is the set of vertices (nodes) of the graph and E is the set of edges. Therefore

$$|V| = N; \quad \text{where } V_i \in V \quad i \in [1, N], \text{ and } d_i = [x_i, \ \ y_i, \ \ t_i] \quad i \in [1, N] \tag{2}$$

For the set of edges E, we assume that each node is connected to k of its immediate neighbours. Therefor E_i is a set of k nodes which are immediate neighbour to V_i and is calculated using the x, and y coordinates. We also define a 3 dimensional input feature vector d_i as defined in the above equation.

2.2 CCN INSPIRED NODE ENCODING

The main function of the encoder is to represent the properties of each graph node and preserving its structural information into a continuous feature vector of dimension d_{embed}, which is digestible in the decoder. Each node i, has three properties which are the x-coordinate, y-coordinate, and the time deadline of the task. The encoding for each node should include its properties and the its positional association with its neighboring nodes. We implement a simple variation of CCN as explained in Hy et al. (2018). We determine the nearest k (here we take $k = 5$) neighbors of a node (Nb_i) based on the positional coordinates (x and y). The first step is to compute a feature vector by linear transformation for each node i.

To encode the node properties, we do a linear transformation of d_i to get a feature vector F_i^d for all $i \in [1, N]$, as shown in Eq. (3).

$$F_i^d = W^d d_i + b_d \tag{3}$$

where $|W^d| = 3 \times d_{\text{embed}}, \ \ |b_d| = 1 \times d_{embed}$. For effective decision making we also need to preserve the structural information. Therefor we define a matrix d_i^{Nb} as defined in Eq. (4).

$$d_i^{Nb} = \text{Concat}(d_j^T)^T, \ \ j \in Nb_i \tag{4}$$

We do a linear transformation of d_i^{Nb} to get F_i^{Nb} (as shown in Eq. (5)), which we believe captures the closeness of a node with its neighbours in terms of the node properties.

$$F_i^{Nb} = W^{Nb}(d_i^{Nb} - d_i) + b^{Nb} \tag{5}$$

where $|W^{Nb}| = 3 \times d_{\text{embed}}, \ \ |b^{Nb}| = 1 \times d_{embed}$. We compute the final embedding for each node using Eq. (6). Here the aggregate function we use is basically the sum of the elements along the column of a matrix. Thus finally we get an embedding F_i for each node, where $|F_i| = d_{\text{embed}}$.

$$F_i = \text{Aggregate}(W_f(\text{Concat}(F_i^d, F_i^{Nb})) + b_f) \tag{6}$$

Here, $|W_f| = d_{\text{embed}} \times d_{\text{embed}}, \ \ |b_f| = 1 \times d_{embed}$. W^d, b_d, W^{Nb}, b^{Nb}, W_f, and b_f are learnable weights and biases.

2.3 ATTENTION-BASED DECODING

Kool et al. (2019) presented an attention mechanism-based model for solving combinatorial optimization problem, where an attention-based encoder and decoder are implemented. Attention mechanism became popular because of it successful implementation in natural language processing. The intuition behind attention mechanism, is to focus on the most relevant part (in terms of language translation). The same concept can be applied for a combinatorial optimization problem with sequential decision making.

The main objective of the decoder is to use the information from the encoder, and the current state as context, choose the best task by calculating a probability value for each node. The first step is feeding the output from the encoder (as key-values) and information from the current state (as context) to a multi-head attention (MHA) layer. The context for the MHA layer in this experiment consist of the following seven features: 1) Current time; 2) Available range of the robot taking decision; 3) Current location of robot taking decision; 4) Current payload delivery capacity; 5) Current destination of

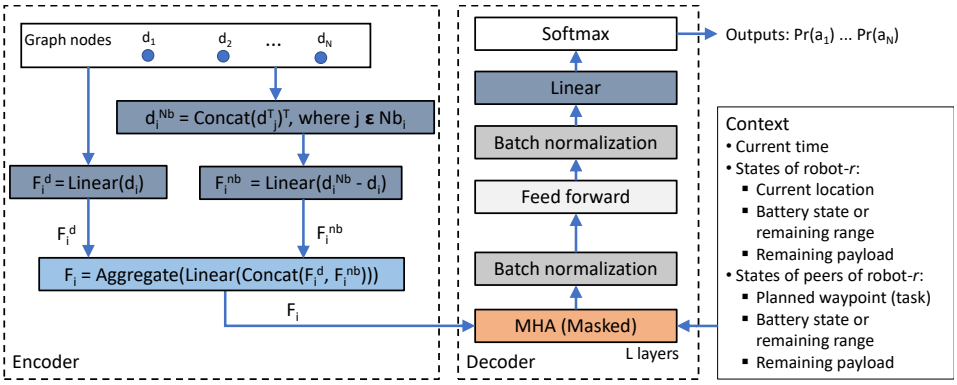


Figure 1: The architecture of the encoder and decoder for CAM that each robot (robot-r) executes. Here, Nb_i, and F_i are the set of neighbours and the final node embedding for node i,

other robots; 6) Available range for other robots; and 7) Current payload delivery capacity for other robots. Figure 1 illustrates the structure of the decoder. Attention mechanism can be described as mapping a query (Q) to a set of key-value (K, V) pairs. The inputs, which are the query (Q), key (K), and values (V), are all vectors. The output is a weighted sum of the values V, weight vector is calculated by the compatibility function:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d^k}})V \tag{7}$$

where d^k is the dimension of K or V. In this work we implement a multi-head attention (MHA) layer inorder to determine the compatibility of Q with K and V. The MHA implemented in this work is the same as the one implemented in (Kool et al., 2019) and (Vaswani et al., 2017). As shown in (Vaswani et al., 2017) the MHA layer can be defined as:

$$\text{Multihead}(Q, K, V) = \text{Linear}(\text{Concat}(\text{head}_1 \dots \text{head}_h)) \tag{8}$$

where $\text{head}_i = \text{Attention}(Q, K, V)$ and h is the number of heads. The feedforward layer is to convert the output from the MHA layer to a dimension that can be taken in for the next process. The final softmax layer outputs the probability values for all the nodes. The next task to be done will be chosen based on a greedy approach, which means the node which has the highest probability will be chosen. The nodes which are already visited will be masked such that it will not be chosen in the future time steps of the simulation.

3 Learning Method

Both the CCN-inspired encoder and the attention-based decoder consist of learnable weight matrix as explained in Sections 2.2 and 2.3. Due to the computational complexity of MRTA (with increasing number of nodes) finding the ground truth for all the training samples will be extremely expensive. Hence we implement a reinforcement learning algorithm. In this work we implement a simple policy gradient method (REINFORCE) as the learning algorithm with greedy rollout baseline, which also enables us to compare the effectiveness of our method with that of (Kool et al., 2019). For each epoch, two sets of data are used which are the training set and the validation set. The training data set is used to train the training model (θ_{CAM}) while the validation data set is used to update the baseline model (θ^{BL}_{CAM}). The size of the training data and the validation data used for this paper is mentioned in Section 4.2. Each sample data from the training and validation dataset consist of a graph as defined in Section 2.1. The pseudo code of the training algorithm for our architecture is shown in Alg. 1.

3.1 Cost Function

The policy gradient method requires the evaluation of a cost function. We hand-craft the cost function such that emphasises on maximizing the completion rate (i.e., the number of completed tasks divided by the total number of tasks), and only if there is a perfect completion rate (100%), then the traveled cost is considered. The cost function is defined as follows:

Algorithm 1 CAM - Training Algorithm

Input: N_{epochs}: Number of epochs, N_b: Number of batch, B: Batch size, N_{tr}: Training data size, N_{vl}: Validation data size.

θ_{CAM} - CAM model
θ_{CAM}^{BL} - Baseline CAM model
for epoch = $1..N_{\text{epoch}}$ **do**
  $\mathcal{D}_{\text{tr}} \leftarrow$ GenerateScenarios(N_{tr})
  $\mathcal{D}_{\text{vl}} \leftarrow$ GenerateScenarios(N_{vl})
  $N_b \leftarrow \lfloor N_{\text{tr}}/B \rfloor$
  for step = $1..N_b$ **do**
    $\mathcal{D}_{\text{tr,b}} \leftarrow$ SampleRandom$(\mathcal{D}_{\text{tr}}, M)$ ▷ $\mathcal{D}_{\text{tr,b}}$: Batch Training Dataset
    $\mathbf{a}^{\text{BL}},\ f_{\text{cost}}^{\text{BL}} \leftarrow$ CalculateCost$(\theta_{\text{CAM}}^{BL}, \mathcal{D}_{\text{tr,b}})$
    $\mathbf{a},\ f_{\text{cost}} \leftarrow$ CalculateCost$(\theta_{\text{CAM}}, \mathcal{D}_{\text{tr,b}})$
    $\nabla\mathcal{L} \leftarrow \frac{1}{B}\sum_{i=1}^{B}(f_{\text{cost,i}} - f_{\text{cost,i}}^{\text{BL}}) \log \text{softmax}(\mathbf{a}_{\text{i}})$
    $\theta_{\text{CAM}} \leftarrow$ ADAM$(\nabla\mathcal{L}, \theta_{\text{CAM}})$
  $\mathbf{a}_{\text{vl}}^{\text{BL}},\ f_{\text{cost,vl}}^{\text{BL}} \leftarrow$ CalculateCost$(\theta_{\text{CAM}}^{BL}, \mathcal{D}_{\text{vl}})$
  $\mathbf{a},\ f_{\text{cost}} \leftarrow$ CalculateCost$(\theta_{\text{CAM}}, \mathcal{D}_{\text{vl}})$
  if $(\sum_{i=1}^{N_{\text{vl}}} f_{\text{cost,i}}^{\text{BL}} > \sum_{i=1}^{N_{\text{vl}}} f_{\text{cost,i}})\ \wedge\ (\text{T-Test}(\mathbf{a}_{\text{vl}}, \mathbf{a}_{\text{vl}}^{\text{BL}}) > \epsilon)$ **then**
    $\theta_{\text{CAM}}^{BL} \leftarrow \theta_{\text{CAM}}$
procedure CALCULATECOST$(\theta, \mathcal{D})$
  $\mathbf{a} \leftarrow [\,]$ ▷ Actions (tour)
  $f_{\text{cost}} \leftarrow [\,]$ ▷ Cost function
  for $i = 1..|\mathcal{D}|$ **do**
    $\mathbf{a}_{\text{i}}, f_{\text{cost,i}} \leftarrow$ Simulation$(\theta, \mathcal{D}_i)$
    $\mathbf{a} \leftarrow \mathbf{a} \cup \mathbf{a}_{\text{i}}$
    $f_{\text{cost}} \leftarrow f_{\text{cost}} \cup f_{\text{cost,i}}$
  return $\mathbf{a}, f_{\text{cost}}$

$$f_{\text{cost}} = r - u(-r)e^{-d_r^2}; \quad \text{where } r = \frac{N - N_{\text{success}}}{N}, \text{and } d_r = \frac{\sum_{i=1}^{N_{\text{r}}} d_i^{\text{total}}}{\sqrt{2}\,N} \tag{9}$$

Here, $u(.)$ is the step function. The term of $1 - r$ is defined as task completion rate; i.e., the number of completed tasks (N_{success}) divided by the total tasks (N). d_r is a normalized value of the total distance traveled by all robots in the team. The terms N_r and d_i^{total} represent respectively the number of robots and the total traveled distance by robot-i during the entire mission. The above cost function (Eq. (9)) gives positive value if the completion rate is lower than 100%, otherwise it gives negative value.

4 Case Studies

4.1 Overview

In this paper, we consider a flood disaster response application, where a team of UAVs deliver survival kits to flood victims. Here, we assume the information on the locations of victims (tasks) is assumed to be available a priori. We design and execute a set of numerical experiments to investigate the performance of our proposed learning-based algorithm over graph space (*CAM*) and compare it with an extended version of a state-of-the-art graph learning-based algorithm proposed by Kool et al. (2019), so called *attention-based* (*AM*) approach. In order to test and validate the approaches, a set of numerical experiments is designed to mimic flood scenarios. The results are compared in terms of the cost function (Eq. (9)), and completion rate.

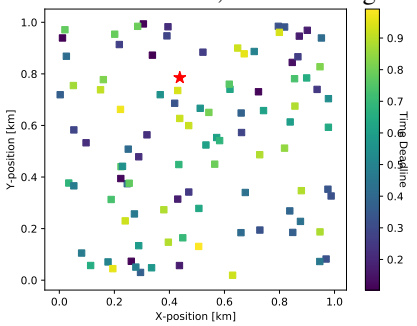


Figure 2: A sample scenario with a depot location (the red star marker) and flood victim locations (the square markers).

4.2 Design of Experiments & Learning Procedures

For this work, we define a case study with 20 UAVs and 200 task locations. The environment is a 2D plane, 1×1 sq.km, and the time deadline of tasks is varying from 0.1 to 1 hour. A sample case scenario is shown in Fig. 2. To perform learning, and testing of the learned model, we are proceeding as follows: *Learning Phase:* We use a policy gradient reinforcement learning algorithm (REINFORCE with rollout baselines in this case) for learning the optimal policy. The learnable parameters in this architecture includes all the weights in the encoder and the decoder. The training is carried out for a total of 200 epochs. Each epoch consists of 10000 random training samples, which are evaluated and trained in batches of 100 samples. *Testing Phase:* In order to provide a statistically insightful evaluation and comparison, 100 random test scenarios are generated for the 20-uav-200-task case study, such that the number of flood victims are constant (200) but their locations and time deadlines are different. We assume instantaneous battery swap and payload replenishment are provided at depot location and when UAVs have exhausted their (survival kit) payloads or are running low on battery, they will need to return to depot. Here, the speed of UAVs is set at 10 km/h.

4.3 Simulation and Framework Settings

The *"Python"* 3.7 and the 64-bit distribution of *"Anaconda 2020.02"* are used to implement the MRTA approaches. The environment, CAM architecture, training algorithm, and the evaluation of the trained model, are all implemented in *Pytorch-1.5*. The attention-based architecture was also implemented in Pytorch. With the help of Pytorch, the training was deployed on two GPUs (NVIDIA tesla v100-pcie-16gb) with 16 GB RAM.

5 Results and Discussion

Learning Curve: In order to compare the sample efficiency of the proposed CAM method with the AM approach, we run both methods with similar settings and plot their learning curve (convergence history), as shown in Fig. 3. As it can be seen from this figure, our primary method, CAM, shows better convergence than the AM method in both terms of the trend and the converged value. The CAM approach converged to 0.029, which is 69.4% better than the converged value of the AM approach. The AM approach reached its final converged value (0.093) after 73 epochs. In contrast, our primary method, CAM, reached the best cost value of AM only after 7 epochs and reached its final converged value after 35 epochs. These results clearly show better sample efficiency of our proposed method, CAM, in comparison with AM. In addition, as it is discussed in below, the computational cost of each epoch for our CAM method is $\sim$2 minutes smaller than the AM method. In overall, our proposed approach reaches better cost function in terms of epoch and much faster in terms of wall time.

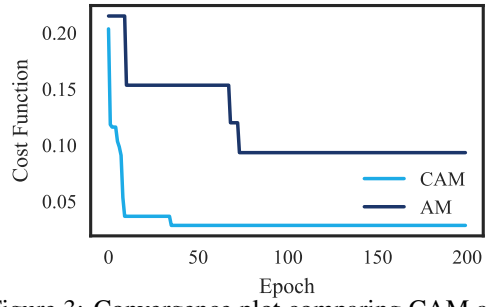
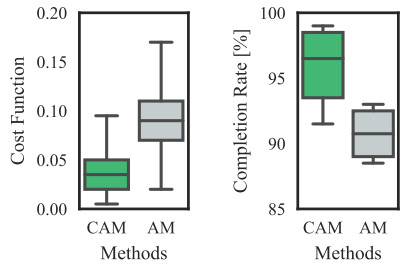


Figure 3: Convergence plot comparing CAM and AM for 200 epochs

Computational Cost Analysis: With the epoch information as described in 4.2, the average time to complete an epoch for training is 12.48 minutes for AM and 10.17 minutes for CAM. This shows the computational cost advantage of our primary method over the AM method. The average decision making time for the trained CAM model is $\sim$2.4 milliseconds while for AM it is $\sim$2.3 milliseconds. The worst performance in terms of decision making computation time for CAM is 5.3 milliseconds, while that for AM is 3.6 milliseconds. These results demonstrate the effectiveness of CAM for decision making (task allocation) in an online setting.

Comparative Analysis: Figure 4 shows the performance of the CAM, and AM approaches for 100 unseen test scenarios in terms of cost function and completion rate. It can be seen from Fig. 4 that our proposed CAM method outperformed the AM approach by achieving 84% and 5% better in terms of the median value of the cost function and the completion rate, respectively. The 84% improvement over the cost function, which is the main metric each approach tries to minimize, clearly shows the superior performance of the proposed method over the AM approach.



(a) Cost function (b) Completion rate

Figure 4: Comparative analysis: The performance of the algorithms over 100 unseen test case scenarios. Each scenario includes 200 tasks and the swarm size is set at 20 robots.

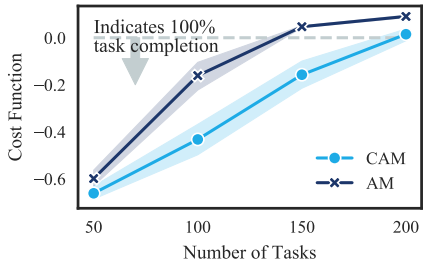


Figure 5: Varying task size for a 20-robot swarm: The performance of the CAM, and AM algorithms over varying task size for 100 unseen test case scenarios per task size. The solid line shows the mean value and the shaded area shows the 95% confidence interval. The dashed gray line shows the threshold value of 0, where any cost below of this line means a 100% completion rate.

Generalisability Analysis: In order to study how the performance of the learnt model is generalizable to new unseen scenarios with different different task size (50 to 200 tasks), a set of experiments are conducted for a 20-robot swarm. For each task size, 100 random scenarios are generated and tested. The results are shown in Fig. 5. In this figure, the dashed horizontal line shows the threshold value of 0, where any cost below that line means a perfect completion rate (100%) and the distance traveled is also considered in the cost function. The performance of both CAM and AM approaches improved by decreasing the number of total tasks. This observation is expected, since less number of tasks means less tasks per robot and easier mission. It can be seen from Fig. 5 that our primary method, CAM, outperforms the AM approach in all case scenarios. The proposed CAM method provided a 100% completion rate for 50-, 100-, and 150-task case studies, while the AM method performed with a 100% completion rate for 50- and 100-task case studies. In terms of the cost function, which is the main objective that each method tried to optimize, the CAM method achieved up to 435% better (i.e., lower value of the cost function) than the AM method.

6 Conclusion

In this paper, we proposed a new graph neural network architecture, called CAM, for a multi-robot task allocation problem with a set of complexities, including tasks with time deadline and robots with ferry range and payload capacity constraints. This new architecture incorporates an encoder based on covariant node-based embedding and a decoder based on attention mechanism. To learn the features of the encoder and decoder, the problem has been imposed as a reinforcement learning problem and it has been solved using a policy simple gradient algorithm, REINFORCE. In addition, to compare the performance of the proposed CAM method, an attention-based approach (aka AM) has been extended to be able to handle a multi-agent combintorial optimization problem (i.e., a multi-robot task allocation problem).

To evaluate the performance of the proposed architecture, and the extended version of AM are trained for 200 epochs and tested on 100 unseen case studies. Performance was analyzed in terms of cost function and completion rate. The new proposed architecture showed a better sample efficiency than AM by reaching a better cost value only after 7 epochs versus 73 epochs of AM. Our primary method, CAM, outperformed the AM approach by achieving (up to) 84% better cost function value (in term of the mean value). The computational cost analysis showed that the proposed CAM model takes a few milliseconds to take a decision; hence, it is an excellent choice for online decision making (here, task allocation). A further study on the performance of both AM and CAM approaches on case studies with different number of tasks demonstrated that our proposed CAM method is superior to AM in all case studies. While the current method is operational over varying task size (the upper bound is the number of tasks that model has been trained on), it is not invariant to the size of swarm. One immediate future work is to expend the architecture to have a swarm size invariant model.

REFERENCES

Jean-Philippe Aurambout, Konstantinos Gkoumas, and Biagio Ciuffo. Last mile delivery by drones: An estimation of viable market potential and access to citizens across european cities. *European Transport Research Review*, 11(1):30, 2019.

Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, 202(3):756–763, 2010.

Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations research*, 59(5):1269–1283, 2011.

Thomas D Barrett, William R Clements, Jakob N Foerster, and Alex I Lvovsky. Exploratory combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1909.04063*, 2019.

Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, 2016.

Diego Cattaruzza, Nabil Absi, and Dominique Feillet. Vehicle routing problems with multiple trips. *4OR*, 14(3):223–259, 2016.

Hanjun Dai, Elias B. Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, 2017.

George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1): 80–91, 1959.

M Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

Maria Valera Espina, Raphael Grech, Deon De Jager, Paolo Remagnino, Luca Iocchi, Luca Marchetti, Daniele Nardi, Dorothy Monekosso, Mircea Nicolescu, and Christopher King. Multi-robot teams for environmental monitoring. In *Innovations in Defence Support Systems–3*, pp. 183–209. Springer, 2011.

Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

Payam Ghassemi and Souma Chowdhury. Decentralized Task Allocation in Multi-Robot Systems via Bipartite Graph Matching Augmented With Fuzzy Clustering. In *Volume 2A: 44th Design Automation Conference*, pp. V02AT03A014. American Society of Mechanical Engineers, aug 2018. ISBN 978-0-7918-5175-3. doi: 10.1115/DETC2018-86161.

Payam Ghassemi, David DePauw, and Souma 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)*, pp. 83–85, New Brunswick, NJ, aug 2019. IEEE. ISBN 978-1-7281-2876-4. doi: 10.1109/MRS.2019.8901062. URL https://ieeexplore.ieee.org/document/8901062/.

Truong Son Hy, Shubhendu Trivedi, Horace Pan, Brandon M Anderson, and Risi Kondor. Predicting molecular properties with covariant compositional networks. *The Journal of chemical physics*, 148(24):241745, 2018.

Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HkxdQkSYDB.

Kelin Jose and Dilip Kumar Pratihar. Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods. *Robotics and Autonomous Systems*, 80:34–42, 2016.

Yoav Kaempfer and Lior Wolf. Learning the multiple traveling salesmen problem with permutation invariant pooling networks. *ArXiv*, abs/1803.09621, 2018.

Nitin Kamra and Nora Ayanian. A mixed integer programming model for timed deliveries in multi-robot systems. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 612–617. IEEE, 2015.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pp. 6348–6358, 2017.

Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. Multi-robot task allocation: A review of the state-of-the-art. In *Cooperative Robots and Sensor Networks 2015*, pp. 31–51. Springer, 2015.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ByxBFsRqYm`.

G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems*, pp. 539–548, 2018.

Lantao Liu and Dylan A Shell. Physically routing robots in a multi-robot network: Flexibility through a three-dimensional matching graph. *The International Journal of Robotics Research*, 32 (12):1475–1494, 2013.

Aristide Mingozzi, Roberto Roberti, and Paolo Toth. An exact algorithm for the multitrip vehicle routing problem. *INFORMS Journal on Computing*, 25(2):193–207, 2013.

Akash Mittal, Anuj Dhawan, Sahil Manchanda, Sourav Medya, Sayan Ranu, and Ambuj Singh. Learning heuristics over large graphs via deep reinforcement learning. *arXiv preprint arXiv:1903.03332*, 2019a.

Akash Mittal, Anuj Dhawan, Sahil Manchanda, Sourav Medya, Sayan Ranu, and Ambuj K. Singh. Learning heuristics over large graphs via deep reinforcement learning. *ArXiv*, abs/1903.03332, 2019b.

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*, 8(4):480–487, 2009.

Alex Nowak, Soledad Villar, Afonso S Bandeira, and Joan Bruna. A note on learning algorithms for quadratic assignment with graph neural networks. *stat*, 1050:22, 2017.

Ernesto Nunes, Marie Manner, Hakim Mitiche, and Maria Gini. A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems*, 90:55–70, 2017.

Edwin Olson, Johannes Strom, Ryan Morton, Andrew Richardson, Pradeep Ranganathan, Robert Goeddel, Mihai Bulic, Jacob Crossman, and Bob Marinier. Progress toward multi-robot reconnaissance and the magic 2010 competition. *Journal of Field Robotics*, 29(5):762–792, 2012.

Eric Schneider, Elizabeth I Sklar, Simon Parsons, and A Tuna Özgelen. Auction-based task allocation for multi-robot teams in dynamic environments. In *Conference Towards Autonomous Robotic Systems*, pp. 246–257. Springer, 2015.

Joseph A Svestka and Vaughn E Huckfeldt. Computational experience with an m-salesman traveling salesman algorithm. *Management Science*, 19(7):790–799, 1973.

Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Di Wang, Mengqi Hu, and Yang Gao. Multi-criteria mission planning for a solar-powered multi-robot system. In *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2018.