Attention Actor-Critic algorithm for Multi-Agent Constrained Co-operative Reinforcement Learning

P. Parnika[*,1], Raghuram Bharadwaj Diddigi[*,2], Sai Koti Reddy Danda[*,3] and Shalabh Bhatnagar[2]

[1] Mindtree Ltd.

[2] Department of Computer Science and Automation, IISc Bangalore, India.

[3] IBM Research, Bangalore, India

parnika.ajay@mindtree.com, {raghub,shalabh}@iisc.ac.in,
saikotireddy@in.ibm.com

Abstract

In this work, we consider the problem of computing optimal actions for Reinforcement Learning (RL) agents in a co-operative setting, where the objective is to optimize a common goal. However, in many real-life applications, in addition to optimizing the goal, the agents are required to satisfy certain constraints specified on their actions. Under this setting, the objective of the agents is to not only learn the actions that optimize the common objective but also meet the specified constraints. In recent times, the Actor-Critic algorithm with an attention mechanism has been successfully applied to obtain optimal actions for RL agents in multi-agent environments. In this work, we extend this algorithm to the constrained multi-agent RL setting. The idea here is that optimizing the common goal and satisfying the constraints may require different modes of attention. By incorporating different attention modes, the agents can select useful information required for optimizing the objective and satisfying the constraints separately, thereby yielding better actions. Through experiments on benchmark multi-agent environments, we show the effectiveness of our proposed algorithm.

Equal contribution by the first three authors. A version of this paper has been accepted for publication as an extended abstract in the Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021).

I. Introduction

In a multi-agent co-operative RL setting [11], multiple agents are working towards a common goal in a common environment. All the agents receive the same cost (or reward) depending on the actions of all the agents and the objective is to minimize (or maximize) the expected total discounted cost (or reward) [30]. However in many practical situations, one often encounters constraints that restrict the choice of actions that can be taken by these agents. In the constrained RL setting [3], these constraints can also be specified via certain expected total discounted costs. In such scenarios, the agents have to learn actions that not only minimize the expected total discounted cost but also respect the constraints.

One approach to satisfy the constraints is to construct a modified cost as a linear combination of the original cost and the constraint costs. However, the weights to be associated with the costs are not known upfront and need to be learned in a trial-and-error fashion. This problem becomes compounded when multiple constraints are specified. We alleviate this problem by considering the Lagrangian formulation of the problem and training dual Lagrange parameters that act as weights for the constraint costs.

References	**Features**
[12], [16], [23], [25], [26]	Deep RL algorithms for multi-agent setting. Attention mechanism not considered.
[20], [21], [24]	Deep RL algorithms with Attention for multi-agent setting. Constrained setting not considered.
[4], [5], [7]	RL algorithms for single-agent constrained setting. Multi-agent constrained setting not considered.
[1], [22], [32]	Deep RL algorithms for single-agent constrained setting. Multi-agent constrained setting not considered.
[2], [9], [14], [15], [17], [27], [29], [34]	RL algorithms for multi-agent constrained setting. Attention mechanism not considered.
Our Work	Deep RL algorithm with Attention mechanism for multi-agent Constrained setting.

Table I: Comparison with other works in the Literature

Single-agent RL algorithms for the constrained RL settings have been proposed under various cost criteria like average cost in [5], [7] and discounted costs in [1], [4], [22], [32]. Constraints in a multi-agent setting can appear in multiple ways. Under budget constraints [9], every joint policy

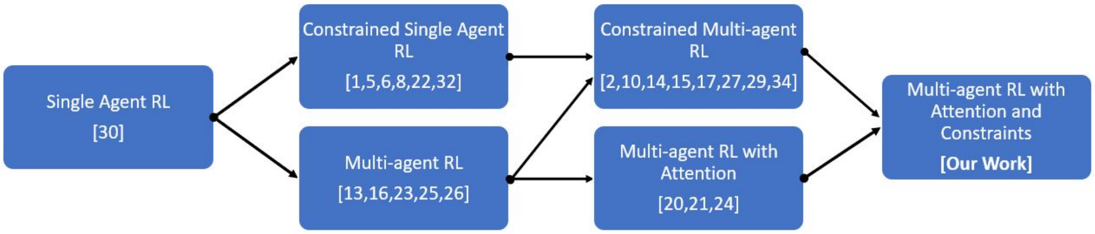

Single Agent RL
[30]
Constrained Single Agent RL
[1,5,6,8,22,32]
Multi-agent RL
[13,16,23,25,26]
Constrained Multi-agent RL
[2,10,14,15,17,27,29,34]
Multi-agent RL with Attention
[20,21,24]
Multi-agent RL with Attention and Constraints
[Our Work]

Figure 1: Evolution of paradigms in literature

is associated with a cost. The objective of the agents here is to compute a joint optimal policy that maximizes the value, respecting the budget constraints. Under resource/task constraints [2], [15], [17], the optimal policy is the one that not only maximizes the value but also optimally allocates the resources to the agents. Under safety constraints [27], [29], [34], each policy is associated with a safety value, and the objective of the agents is to compute optimal policies that meet the safety constraints. Finally, in [14], similar to the model we consider in this paper, the constraints are specified as expected discounted cost which are required to be less than a prescribed threshold value.

Actor-Critic algorithms [30] are a popular class of RL algorithms that are used by an agent to obtain an optimal policy. In this paradigm, 'Actor' computes the policy and 'Critic' provides feedback on the policy computed by the 'Actor'. Based on this feedback, 'Actor' improves the policy. This process is repeated until an optimal policy is obtained. The Actor-Critic paradigm for multi-agent settings can be extended in three ways [26]. All the agents can independently (without co-operation and communication) run the Actor-Critic algorithm. This setting of agents is known as 'Independent Learners' [31] and it suffers from the problem of non-stationarity [11]. Another setting known as 'Joint Action Learners' assumes the existence of a central controller which computes the optimal policy of all the agents and communicates the actions to the agents. This setting suffers from scalability problems as the state and action spaces for the central controller increase exponentially as the number of the agents increases. Finally, a paradigm that mitigates the problem of scalability and non-stationarity known as 'centralized learning and decentralized execution' has become popular in recent times [12], [16], [23], [25]. The main idea here is to use a centralized critic during the training and decentralized actors that learn actions independently. In these algorithms, however, information of all the agents are given equal importance (or weights) while computing the optimal policy.

The attention mechanism allows an agent to selectively pay attention to those agents whose information is crucial in the computation of its policy. In [20], attention actor-critic algorithms have been proposed that make use of the attention mechanism in the learning of 'critic'. In [24], attention mechanism has been used to model the policies of teammates. The attention mechanism for learning communication among the agents has been proposed in [21]. In this work, the authors propose an attentional communication model ATOC that provides an effective mechanism for communication among agents resulting in better decision making. We illustrate the advantages of the attention mechanism through the following example. Consider a Smart Grid setup [28] where a network of microgrids are employed, whose objective is to provide power to its dedicated customers. These microgrids are equipped with renewable energy generation sources (like solar panels, wind turbines) and limited storage battery device to store their power. At every time instant, each microgrid has to make intelligent decisions like the number of units of power to store in its battery, the number of units of power to buy (or sell) from (to) other microgrids to maximize its profits. For any given microgrid, the state information of its neighboring microgrids is more important than those at a far distance from it. Through the attention mechanism, a microgrid can dynamically select these neighboring microgrids, instead of attending to all the microgrids equally. This results in better decision making and hence better profits for the microgrids.

We believe that the attention mechanism is particularly important in the constrained multi-agent setting. We explain its importance through the two following examples.

1) Consider a warehouse where multiple robots are deployed. The objective of the robots is to pick up the goods from a target position with a constraint that the expected number of collisions among the robots is less than a predefined threshold. The important information for a robot for collecting goods is the position of goods whereas the relative distance between the robots is crucial information to avoid collisions. Hence having attention mechanisms separately for learning optimal actions to collect the goods and avoid collisions will be natural in this setting.
2) In the smart grid setting considered, let's say that we impose a constraint that the expected demand-supply deficit should be maintained at a certain level (to ensure the stability of the grid). The relevant information for maximizing the profits and maintaining the stability for a microgrid can be different. The attention mechanism enables the microgrid to attend to the relevant information for these two tasks separately.

Moreover, the attention mechanism for multi-agent constrained setting finds its applications in numerous settings like self-driving cars [29] to ensure safety constraints, supply chain optimization [18] to ensure the resource constraints. In our work, we propose an attention-based Actor-Critic algorithm for solving the problem of multi-agent constrained Reinforcement Learning (RL). While the attention mechanism and constrained RL settings have been studied extensively in the literature, the use of two separate attention mechanisms for computing the policy and satisfying the constraints has not been considered previously. We believe that this architecture is very important as optimizing the common goal and satisfying the constraints require different modes of attention. We show through our analysis of attention weights that using multiple attentive critics can benefit and yield much better results on complicated real-world applications. A brief overview of the comparison of our work with other works in the literature is provided in Table I and the evolution diagram of these paradigms/themes is shown in Figure 1. The main contributions of our work are the following:

- We propose an Actor-Critic algorithm for computing the optimal actions for agents in a constrained co-operative multi-agent setting that makes use of the attention mechanism.
- We analyze and discuss the performance of our algorithm on constrained versions of standard multi-agent RL environments.
- We provide a detailed analysis of the attention mechanism learned by the agents in our experiments (Section IV-B3).

The rest of the paper is organized as follows. In Section II, we describe the multi-agent constrained co-operative setting considered in the paper. In Section III, we propose our multi-agent attention mechanism-based constrained Actor-Critic algorithm. In Sections IV and V, we present the performance of our algorithm on multi-agent environments and discuss the results. Concluding remarks are given in Section VI.

II. Model

We now discuss the constrained co-operative multi-agent setting described in [13]. It is described by tuple $< n, S, A, T, k, c_1, \ldots, c_m, \gamma >$. Here, n denotes the number of agents in the environment. $S = S_1 \times S_2 \times \ldots S_n$ is the joint state space and $s \in S = (s_1 \ldots s_n)$ is the joint state with $s_i \in S_i$ being the state of the agent i. Similarly, $A = A_1 \times, \ldots, \times A_n$ denotes the joint action space where $a \in A = (a_1, \ldots, a_n)$ is the joint action and $a_i \in A_i$ being the action of agent i. Each agent only observes its own state and chooses its action based on it.

Let T be the probability transition matrix where $T(s'|s,a)$ denotes the probability of next state being s' when joint action a is taken in joint state s. Single-stage cost function (k) is the cost incurred when joint action a is taken in state s. Moreover, $c_1,\ldots,c_m$ denote the single-stage cost functions for the constraints. Note that both the main cost function (k) and constraint costs $(c_1,\ldots,c_m)$ depend on the joint action of the agents. Finally, γ denotes the discount factor. Let $\pi_i : S_i \rightarrow \Delta(A_i)$ denote the policy of agent i, where for a given state of agent i, $\pi_i(s_i)$ is a probability distribution over its actions. We now define the total discounted cost (J) for a joint policy $\pi = (\pi_1,\ldots,\pi_n)$ as follows:

$$J(\pi) = E\Big[\sum_{t=0}^{\tau} \gamma^t k(s_t, \pi(s_t))\Big], \tag{1}$$

where $E(.)$ is the expectation over entire trajectory of states with initial state $s_0 \sim d_0$, where d_0 is a probability distribution over states, τ is a finite stopping time and s_t is the joint state at time t. The m constraints on the system are defined as follows:

$$E\Big[\sum_{t=0}^{\tau} \gamma^t c_j(s_t, \pi(s_t))\Big] \leq \alpha_j, \ \forall j \in 1,\ldots,m, \tag{2}$$

where $\alpha_1,\ldots,\alpha_m$ are pre-specified thresholds.

The objective of the agents in the multi-agent constrained co-operative RL setting is to compute a joint policy $\pi^* = (\pi_1^*,\ldots,\pi_n^*)$ that

$$\begin{aligned} \min_{\pi\in\Pi} \ J(\pi) = \ & E\Big[\sum_{t=0}^{\tau} \gamma^t k(s_t, \pi(s_t))\Big] \qquad (3) \\ \text{s.t} \ \ & E\Big[\sum_{t=0}^{\tau} \gamma^t c_j(s_t, \pi(s_t))\Big] \leq \alpha_j, \ \forall j \in 1,\ldots,m, \end{aligned}$$

where Π is set of all joint policies. The constrained problem (3) can be relaxed using the Lagrangian formulation [4], [7] as follows:

$$L(\pi,\lambda) = E\Big[\sum_{t=0}^{\tau} \gamma^t \big(k(s_t,\pi(s_t)) + \sum_{j=1}^{m} \lambda_j c_j(s_t,\pi(s_t))\big)\Big] - \sum_{j=1}^{m} \lambda_j \alpha_j, \tag{4}$$

where $\lambda = (\lambda_1,\ldots,\lambda_m)$ is the vector of Lagrange parameters associated with the m constraints.

From the theory of duality in optimisation (Chapter 5 of [10]), it is clear that the optimal policy π^* and Lagrange parameters λ^* satisfy the following:

$$L(\pi^*,\lambda^*) = \max_{\lambda>0} \min_{\pi\in\Pi} L(\pi,\lambda). \tag{5}$$

The theory of two time-scale stochastic approximation [6] allows us to iteratively learn the Lagrange parameters and policy. The main idea is to perform gradient descent on the objective (4) in the space of policy parameters on the faster timescale and gradient ascent on (4) in the Lagrange parameters on the slower timescale [8]. The complete details of how we achieve this is described in the next section.

III. Proposed Algorithm

Attention mechanism [24], [33]: In general, the attention mechanism works as follows. It takes source vectors $v = (v_1, \ldots, v_n)$ and a target vector T as inputs and outputs a context vector C. The attention mechanism first computes attention weights $(w_1, \ldots, w_n)$, where $w_i,\ 1 \leq i \leq n$ represents the importance of v_i. The attention weight w_i is computed from a given function $f(T, v)$ as follows:

$$w_i = softmax(f(T, v)) = \frac{exp(f(T, v_i))}{\sum_{j=1}^{n} exp(f(T, v_j))}. \tag{6}$$

Finally, the context vector is computed as:

$$C = \sum_{j=1}^{n} w_j v_j. \tag{7}$$

As it can be seen from (6) that $\sum_{j=1}^{n} w_j = 1$, the attention mechanism can thought of a computation that adaptively learns the distribution over input vector that accurately represent the context of the problem.

We extend the attention mechanism proposed in the context of the multi-agent RL setting [20] to the constrained setting. The details of the proposed attention mechanism are as follows. Each agent i maintains a total of $m+1$ critics which use attention. Let's denote these as the cost critic Q_ψ (associated with the main cost function) and m penalty critics $Q_{\eta 1}, \ldots, Q_{\eta m}$ (associated with the m constraints). The intuition here is, by having multiple critics with different attentions, each critic is especially able to attend to that information which is crucial in solving its objective. The way this information is utilized for attention is by encoding state and state-action information of all agents where the embedding function is a single layer perceptron.

These encodings are passed to another embedding function, also a single layer perceptron, to create keys (K), values (V), and selectors/queries (q) [20]. The keys K_j and values V_j represent state-action encodings of all agents $j \neq i$ while queries q_i are state encodings of the agent i. Now, the attention weights w_j are computed as a function of queries and keys as follows:

$$w_j = softmax\Big(\frac{q_i K_j^T}{\sqrt{d_k}}\Big), \tag{8}$$

where d_k is the size of the keys. Finally, critic Q of agent i (denoted by Q^i) is obtained as follows (for notation convenience, we drop the subscripts from the critics of agent i as all $m+1$ critics use similar architecture):

$$Q^i = f_i(g_i(o_i, a_i), x_i), \tag{9}$$

where f_i is a multi-layer perceptron with two layers, g_i is an embedding function for agent i, and $x_i = \sum\limits_{j \neq i} w_j V_j$ is the contribution of other agents to agent i.

We now discuss our proposed algorithm 'MACAAC' (Algorithm 1). We train the algorithm in μ parallel environments to improve the sample efficiency and reduce the variance of updates. At each time step of an episode, every agent samples an action from its current policy based on its observations o_i and obtains common single-stage cost k, single-stage penalties $c_1, \ldots, c_m$ and next state as shown in steps 19 of Algorithm 1. The Lagrangian cost is calculated as shown in the step 20. This information is then stored in the replay buffer D. The 'Critic' and 'Actor' parameters are updated after every U steps. This is done as follows. First a minibatch 'B' is sampled independently from the replay buffer. For each sample from the minibatch 'B', the critic parameters are updated (Step 27 of Algorithm 1) by performing gradient descent on the MSE loss given by [20]:

$$\sum_{i=1}^{n} E[(Q^i(o, a) - y_i)^2], \tag{10}$$

where $y_i = r + \gamma E[Q^i(o^*, a^*) - \alpha \log(\pi_\theta(a_i^* | o_i^*))]$, o^*, a^* are the joint next state and actions of agents and α is known as the temperature coefficient [20] that is used to control the stochastic nature of the policy. The parameters of cost critic ψ are updated by performing gradient descent on (10) with r defined as in Line 20 of Algorithm 1 (Lagrangian cost) and the parameters of penalty critic j, η_j, are updated by performing gradient descent on (10) with r as c_j (constraint cost). Moreover, all the agents i share the same parameters $(\psi, \eta_1, \ldots, \eta_m)$ of critics.

The 'UpdateActors' step is performed as follows. The policy parameters of each agent (θ_i) are updated by performing gradient descent using the gradient function given by [20]:

$$E[\nabla_{\theta_i} \log(\pi_{\theta_i}(a_i|o_i))(-\alpha \log(\pi_{\theta_i}(a_i|o_i)) + Q_\psi^i(o, a) - b(o, a_{-i}))], \tag{11}$$

where b is a baseline function that is independent of actions of agent i (a_{-i} represents the actions of all agents except i). Note that in equations (10) and (11), an entropy term is added that facilitates stochastic policies [19].

Finally, the Lagrange parameters $\lambda_j,\ j = 1, \ldots, m$ are updated by performing gradient ascent on the Lagrangian L (eq. 4) as shown in the step 30. An important point to note here is that the critic and actor updates (steps 27 and 28) are performed on a faster time-scale compared to the Lagrange parameter updates (steps 29-30). As a result, the critic and actor perceive Lagrange parameters as constants in their updates, thereby ensuring the convergence of the algorithm [8, Chapter 6].

IV. Experiments and Results

In this section, we describe the performance of our proposed Algorithm 'MACAAC' on two multi-agent environments and analyze the results. The first environment is the constrained version of Cooperative Navigation [23] followed by the constrained version of Cooperative Treasure Collection [20]. The constraint considered in the experiments is the collision between the multiple agents. The agents incur a penalty whenever there is a collision and their objective is to make sure that the expected total penalty is less than a prescribed penalty threshold. To avoid confusion, we refer to the main cost that the agents are minimizing as 'cost' and the constrained cost as the 'penalty'. For comparison purposes, we also implement the constrained version of MADDPG [23] algorithm, which we refer to as 'MADDPG-C'. Moreover, to better analyze the results, we also report the results on an un-constrained version of Multi-agent Attention Actor-Critic [20] where there is no penalty incurred for collisions among the agents, which we simply refer to as 'Unconstrained'. Finally, in section V, we evaluate the performance of 'MACAAC' with fixed weights. The neural network architecture and hyper-parameters are kept the same for all three algorithms [1]

A. Constrained Cooperative Navigation

1) Description of the experiment: In this experiment, there are 5 agents and 5 targets that are randomly generated in a continuous environment at the beginning of each episode as shown in the Figure 2. The objective of the agents is to navigate towards the targets in a co-operative

[1] The source codes of our experiments are available at: https://github.com/parnika31/MACAAC_Supplementary

Algorithm 1 Multi-Agent Constrained Attention Actor-Critic (MACAAC)

```
$E \leftarrow$ Maximum number of episodes.
$L \leftarrow$ Length of an episode.
$U \leftarrow$ Steps per update.
$\theta_i \leftarrow$ policy parameters of the agent $i$, $i = 1, \dots, n$.
**UpdateCritic:** Subroutine to update the critic parameters.
**UpdateActors:** Subroutine to update the policy parameters of all the agents.
$Q_{\eta_j} \leftarrow$ Q-value of constrained cost associated with constraint $j$, $j = 1, \dots, m$.
$\beta_t \leftarrow$ Slower timescale step-size at time step $t$.
Initialize Lagrange parameters $\lambda_1, \dots, \lambda_m$.
Create $\mu$ parallel environments.
Initialise replay buffer, D.
$u \leftarrow 0$
**for** $ep = 1, 2, \dots, E$ **do**
    Obtain initial observations $o_i^e$ for all agents $i$ in each
    environment $e$
    **for** $t$ = $1, 2, \dots, L$ **do**
        Obtain actions $a_i^e \sim \pi_{\theta_i}(.|o_i^e)$, $\forall i = 1, \dots, n$,
        $\forall e = 1, \dots, \mu$
        Execute actions and get $(o_i^{*,e}, k^e, c_1^e, c_2^e, \dots, c_m^e) \quad \forall i, e$
        Let $r^e = k^e + \sum_{j=1}^{m} \lambda_j c_j^e, \quad \forall e$
        Store $(o_i^e, a_i^e, r^e, c_1^e, c_2^e, \dots, c_m^e, o_i^{*,e})$, $\forall i, e$ in $D$
        $o_i^e = o_i^{*,e}$, $\forall i, e$
        $u+ = \mu$
        **if** $(u\%$ U) ¡ $\mu$ **then**
            Sample minibatch (B) from D
            Get next actions $a_1^{'}, \dots, a_n^{'}$
            **UpdateCritic**(B, $a_1^{'}, \dots, a_n^{'}$)
            **UpdateActors**(B)
            **for** $j = 1, \dots, m$ **do**
                $\lambda_j \leftarrow \max(0, \lambda_j + \beta_t(Q_{\eta_j} - \alpha_j))$
```

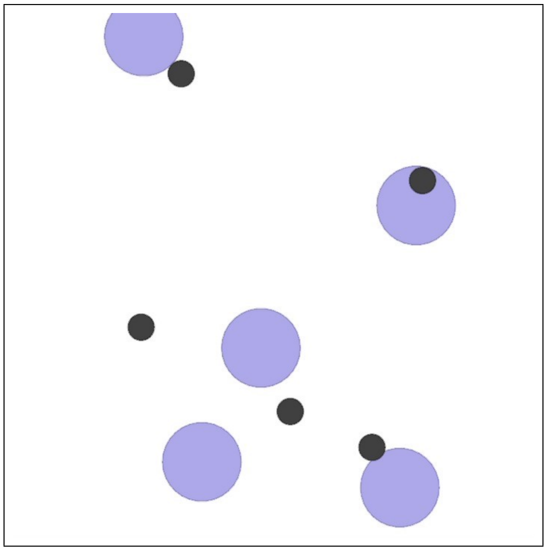


Figure 2: Constrained Cooperative Navigation. The large blue balls are ‘agents’, whose objective is to navigate towards the small black balls which are ‘targets’ without collisions.

manner such that all targets are covered. The length of each episode is 25 time steps and the single-stage cost at each time step is the sum of the distance to the nearest agent, over all the targets. Therefore, the agents have to learn to navigate towards the targets in such a way that all target positions are covered. However, we include a single-stage penalty of 1 when there is a collision between the agents (and 0 otherwise). The penalty threshold (α) is set to 3 in our experiments. This means that the expected total penalty over all the episodes must be less than or equal to 3. The discount factor is set to 0.99. In Figure 4, we show the performance of algorithms during the training phase, and in Table II, we report the performance of algorithms during the testing.

2) Discussion:

- In Figure 4a, we observe that the total cost approaches convergence for all the three algorithms. The ‘Unconstrained’ algorithm achieves the smallest average cost as there is no penalty for collisions in this case. Therefore, the agents can move freely in the continuous space and navigate quickly towards the targets. This can also be observed in Figure 4b, where we see that the average penalty of the ‘Unconstrained’ algorithm is the highest.
- In Figure 4b, we see that the average penalty comes down as the training progresses for the constrained algorithms (MADDPG-C and our proposed MACAAC), while for the ‘unconstrained’ algorithm it almost remains constant. This is the effect of Lagrange parameters

that are learnt in the constrained setting.

- From Table II, we can see that both our proposed algorithm ‘MACAAC’ and ‘MADDPG-C’ satisfies the penalty constraint. However, our algorithm ‘MACAAC’ achieves this average penalty at a lower cost than the ‘MADDPG-C’ algorithm.

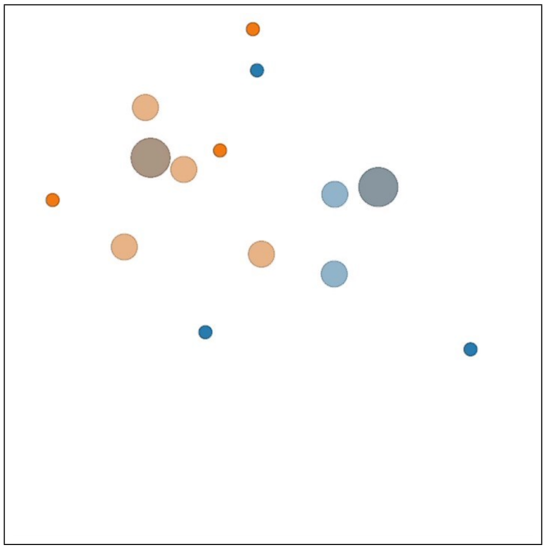
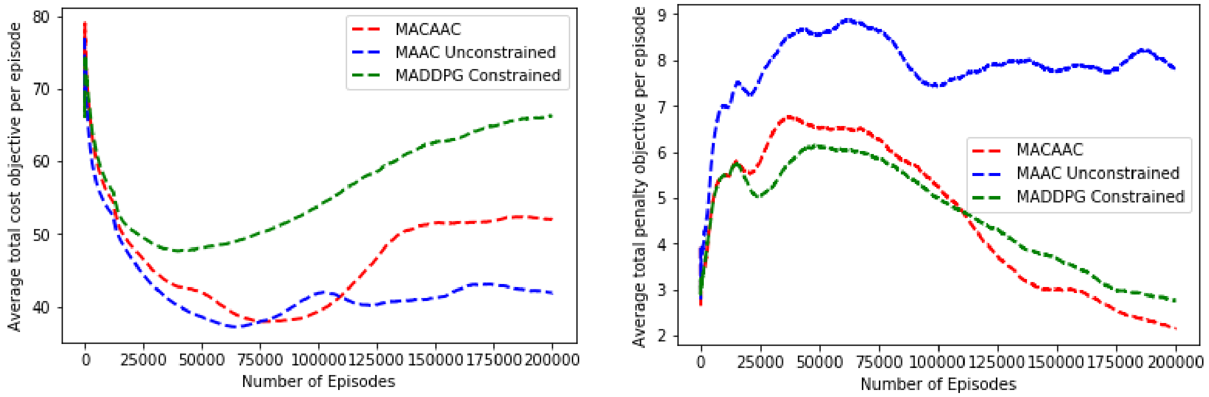


Figure 3: Constrained Cooperative Treasure Collection. The big blue and brown balls are ‘depositors’. The dark orange and blue colored balls are ‘treasures’ which will be re-spawned after every capture. The rest of the balls are ‘collectors’. In this figure, the collector agents changed color after capturing the treasures and are moving towards depositors (or banks) of same color to deposit them without collisions.

B. Constrained Cooperative Treasure Collection

1) Description of the experiment: In this experiment, we have a total of 8 agents, out of which 6 agents are ‘collectors’ (Agents $1, \ldots, 6$, and the other two are ‘deposits (or banks)’ (Agents 7 and 8). The role of collectors is to collect the ‘treasures’ that are randomly generated in the environment and deposit them into the ‘banks’ of the same color as the treasure. New treasures will be re-generated once the existing treasures are collected. The role of the ‘depositors’ is to stay close to the collectors carrying their treasures. The length of each episode is 100 time-steps where all agents receive the shared single-stage cost associated with the total distances from their goals. Moreover, a cost of -5 (a positive reinforcement) is added every time a treasure is



(a) Expected total cost

(b) Expected total penalty

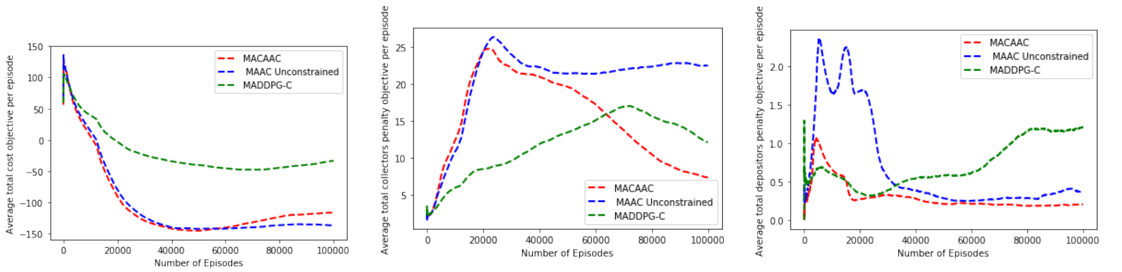
Figure 4: Performance of Algorithms on Constrained Cooperative Navigation during the training. The average total cost and penalty at each episode i are calculated, by taking mean of total costs and total penalties over 1024 runs, using the policies trained until i episodes.

Name of the Algorithm	Average total cost over $10,000$ episodes	Average total penalty over $10,000$ episodes
MACAAC	**45.79**	**1.87**
MADDPG-C	60.33	2.52
Unconstrained	37.50	7.02
MACAAC with Fixed Weights	38.73	1.25

Table II: Performance comparison of algorithms in testing phase on Constrained Cooperative Navigation with penalty threshold α= 3. The average total cost and penalty are calculated by taking mean of total costs and penalties, respectively, over $10,000$ runs using the policies of agents obtained at the end of training.

collected and deposited[2]. We consider two penalty constraints in this experiment for collectors and depositors separately to demonstrate the effect of attention weights (discussed in Section IV-B3). The penalty threshold for collectors ($\alpha 1$) is set to 12 and depositors ($\alpha 2$) is set to 0.2 and the discount factor is 0.99.

[2] This is the reason the costs in Table III are negative, as the agents learn to collect and deposit treasures



(a) Expected total cost

(b) Expected total penalty of collectors

(c) Expected total penalty of depositors

Figure 5: Performance of Algorithms on Constrained Cooperative Treasure Collection during the training.

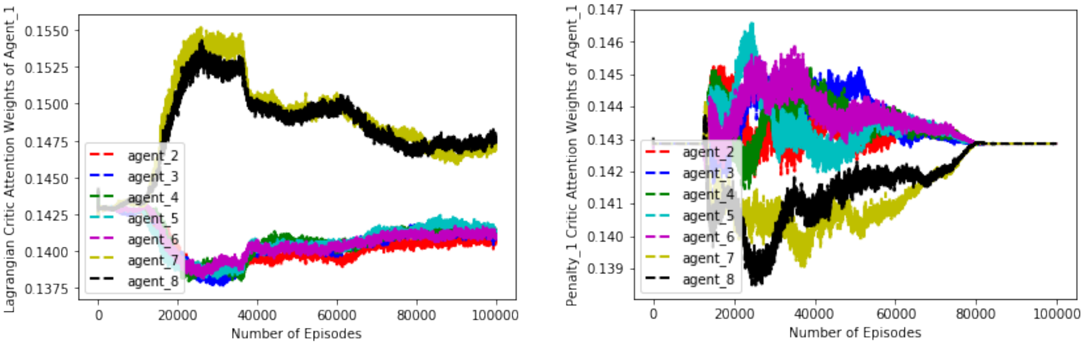
Name of the Algorithm	Average total cost over $10,000$ iterations	Average total penalty of collectors over $10,000$ iterations	Average total penalty of depositors over $10,000$ iterations
MACAAC	**-76.21**	**4.70**	**0.15**
MADDPG-C	-22.20	7.59	0.76
Unconstrained	-88.41	13.99	0.35
MACAAC with Fixed Weights	-54.68	2.63	0.18

Table III: Performance comparison of algorithms in testing phase on Constrained Cooperative Treasure Collection with penalty threshold for collectors ($\alpha 1$) set to 12 and that of the depositors ($\alpha 2$) set to 0.2.

2) Discussion:

- As in our previous experiment, we can observe from Figure 5a that the average total cost converges for all three algorithms.
- From Table III, we can see that our proposed ‘MACAAC’ satisfies the penalty constraints of both ‘collectors’ and ‘depositors’. Moreover, the average cost obtained by ‘MACAAC’ is lower compared to the ‘MADDPG-C’ algorithm. As the agents do not incur a penalty in ‘Unconstrained’, its average cost is least among three algorithms.

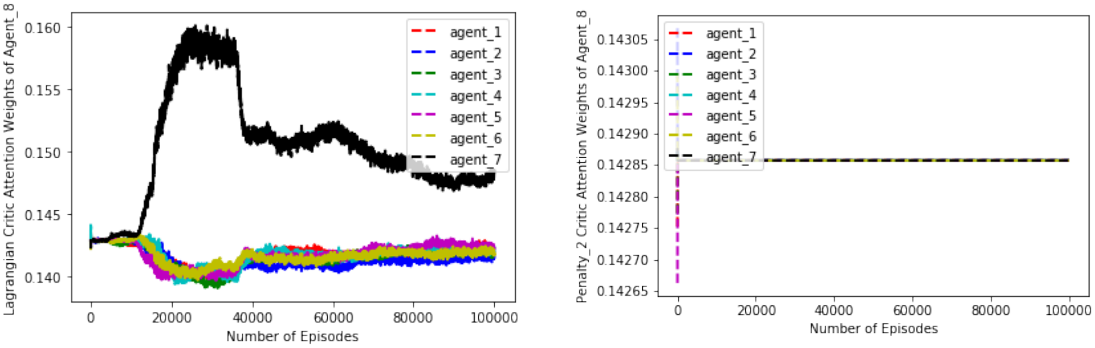
3) Discussion of Attention graphs: We now discuss the attention weights learned by the agents during the training. We present the attention weights learnt by the agent 1, which is a ‘collector’ in Figure 6 and agent 8, which is a ‘depositor’ in Figure 7. Recall that, there are six collectors



(a) Attention Weights of Lagrangian critic

(b) Attention weights of penalty_1 critic

Figure 6: Attention weights of Agent 1. These plots indicate the attention weights assigned to other agents by Agent 1



(a) Attention Weights of Lagrangian critic

(b) Attention weights for penalty_2 critic

Figure 7: Attention weights of Agent 8. These plots indicate the attention weights assigned to other agents by Agent 8

(agents 1-6) and two depositors (agents 7 and 8). In this experiment, there are three critics that use attention. Two penalty critics, which we refer to as penalty_1 and penalty_2 critics, compute the expected penalty costs of collectors and depositors respectively. The feedback from these critics is used in improving the Lagrange parameters (Step 30 of Algorithm 1). Then, there is the main critic whose feedback is used to improve the policy parameters (Step 28 of Algorithm 1). Note that the penalty critics make use of only the penalty costs whereas the main critic (Lagrangian critic) makes use of Lagrangian cost that involves both main cost and penalty costs (Step 20 of Algorithm 1).

In Figure 6a, we observe that the Lagrangian Critic of agent 1 focuses more on the depositors[3] throughout its training. The agent 1 required to deposit its collected treasures into depositors, to minimize its cost, and hence the Lagrangian critic attends more to information of the depositors. The attention graph of Penalty_1 critic of agent 1 in Figure 6b is very interesting. At the beginning of training, agent 1, to avoid collisions, focuses more on the information of other collectors and less on the depositors. However, as the training progresses, all the agents learn to move towards the depositors to deposit their treasures. Hence, the information of the depositors becomes very relevant for agent 1 to avoid collisions. This can also be confirmed from Figure 5b where the constraint ($\alpha = 12$) is being satisfied, towards the end, after $80k$ iterations. Therefore, we observe that, towards the end of the training, agent 1 attends to information of all the other agents equally. In this way, the attention mechanism enables the agents to dynamically select relevant information during the training.

In Figure 7, we report the attention graphs of agent 8, which is a depositor. In Figure 7a, we observe that agent 8 attends more to the information of other depositors, i.e., agent 7. We have seen earlier that the collectors attend more to the information of the depositors to deposit their treasures. Moreover, the Lagrangian cost is a combination of the main cost and penalty costs. Therefore, agent 8 has to move in directions that don't overlap with agent 7, thereby providing collectors enough space to safely (avoiding collisions) deposit their treasures. Finally, in Figure 7b, we see that penalty_2 critic of agent 8 attends to information of all the agents uniformly throughout its training. Similar to the penalty_1 critic of agent 1, information of all the agents is equally important for the agent 8 to avoid collisions.

In this way, our proposed algorithm provides a framework for multi-agents to learn suitable attentions for various sub-tasks. The advantage of this paradigm can be seen from our results, where our proposed algorithm 'MACAAC' performs well while satisfying the specified penalty constraints.

V. Effect of Fixed weights for Constrained Costs

As discussed in the introduction section, a constrained problem could be solved by adding the constrained costs to the main cost. However, the weights to be associated with the constrained costs to satisfy the specified constraints are not known. Therefore, in our proposed algorithm, on

[3] as higher attention weights are assigned to agents 7 and 8

a slower time-scale, Lagrangian parameters are iteratively learnt, which act as weights for the constrained costs. In this section, we investigate the effect of using constant and fixed weights for the constrained costs during the training. That is, we construct a cost function as $k+\sum_{j=1}^{m} w_j c_j$, where k is the main cost function, $c_j,\ 1 \leq j \leq m$ are m constrained costs and w_j is the weight associated with constraint j. The weights we use in the experiments are the converged Lagrange parameters from the "MACAAC" algorithm. We call this experiment "MACAAC with Fixed Weights".

In "Constrained Cooperative Navigation," there is one constraint and the weight assigned to this constraint is $w_1 = 5.534$. We observe from Table II that, this value of w_1 satisfies the penalty constraint $\alpha = 3$. Moreover, the average cost obtained is slightly less than that of the standard "MACAAC".

In Table III, we run the "MACAAC with Fixed Weights" for the "Constrained Treasure Collection" experiment. The weights assigned to two penalty constraints in this experiment are $w_1 = 3.47$ and $w_2 = 0.83$. Similar to our earlier experiment, we observe that these weights satisfy the penalty constraints of both collectors and depositors. However, the average cost obtained is higher than the standard "MACAAC" algorithm as these fixed weights may be too restrictive in this experiment. On the other hand, our proposed algorithm adaptively trains (increases or decreases) the Lagrange parameters during the training, leading to a better policy.

From this study, we conclude the following:

1) Our proposed "MACAAC" adaptively computes the Lagrange parameters that satisfy the penalty constraints.
2) Our proposed "MACAAC" algorithm computes a near-optimal solution using a two-time scale approach, where policy is updated on a faster timescale and Lagrange parameters are updated on a slower timescale.

VI. Conclusions

We have considered a constrained multi-agent RL setting where the agents need to learn optimal actions that satisfy the constraints specified on their actions. We have proposed an attention mechanism based constrained Actor-Critic algorithm that computes the Lagrange parameters on a slower time-scale and optimal policy on a faster time-scale. The attention mechanism enables the agents to select relevant information during the training, for computing the policy and satisfying the constraints. Through experiments on two benchmark multi-agent settings, we have shown that our proposed algorithm computes a near-optimal solution satisfying the penalty constraints.

VII. Acknowledgements

Raghuram Bharadwaj was supported by a fellowship grant from the Centre for Networked Intelligence (a Cisco CSR initiative) of the Indian Institute of Science, Bangalore. This work was supported by the Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science, and a grant from the Department of Science and Technology, India. S.Bhatnagar was also supported by the J.C.Bose Fellowship.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. *arXiv preprint arXiv:1705.10528*, 2017.

[2] Pritee Agrawal, Pradeep Varakantham, and William Yeoh. Scalable greedy algorithms for task/resource constrained multi-agent stochastic planning. 2016.

[3] Eitan Altman, Konstantin Avrachenkov, Richard Marquez, and Gregory Miller. Zero-sum constrained stochastic games with independent state processes. *Mathematical Methods of Operations Research*, 62(3):375–386, 2005.

[4] Shalabh Bhatnagar. An actor–critic algorithm with function approximation for discounted cost constrained markov decision processes. *Systems & Control Letters*, 59(12):760–766, 2010.

[5] Shalabh Bhatnagar and K Lakshmanan. An online actor–critic algorithm with function approximation for constrained markov decision processes. *Journal of Optimization Theory and Applications*, 153(3):688–708, 2012.

[6] Vivek S Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5):291–294, 1997.

[7] Vivek S Borkar. An actor-critic algorithm for constrained markov decision processes. *Systems & control letters*, 54(3):207–213, 2005.

[8] Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.

[9] Craig Boutilier and Tyler Lu. Budget allocation using weakly coupled, constrained markov decision processes. 2016.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[11] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*, 2008.

[12] Gang Chen. A new framework for multi-agent reinforcement learning–centralized training and exploration with decentralized execution via policy distillation. *arXiv preprint arXiv:1910.09152*, 2019.

[13] Raghuram Bharadwaj Diddigi, Sai Koti Reddy Danda, Prabuchandran K.J., and Shalabh Bhatnagar. Actor-critic algorithms for constrained multi-agent reinforcement learning. *arXiv preprint arXiv:1905.02907*, 2019.

[14] Raghuram Bharadwaj Diddigi, D Reddy, Prabuchandran KJ, and Shalabh Bhatnagar. Actor-critic algorithms for constrained multi-agent reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1931–1933. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[15] Dmitri A Dolgov and Edmund H Durfee. Resource allocation among agents with mdp-induced preferences. *arXiv preprint arXiv:1110.2767*, 2011.

[16] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926*, 2017.

[17] Michael Fowler, Pratap Tokekar, T Charles Clancy, and Ryan K Williams. Constrained-action pomdps for multi-agent intelligent knowledge distribution. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[18] Ilaria Giannoccaro and Pierpaolo Pontrandolfo. Inventory management in supply chains: a reinforcement learning approach. *International Journal of Production Economics*, 78(2):153–161, 2002.

[19] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[20] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2961–2970, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[21] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, pages 7254–7264, 2018.

[22] Qingkai Liang, Fanyu Que, and Eytan Modiano. Accelerated primal-dual policy optimization for safe reinforcement learning. *arXiv preprint arXiv:1802.06480*, 2018.

[23] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[24] Hangyu Mao, Zhengchao Zhang, Zhen Xiao, and Zhibo Gong. Modelling the dynamic joint policy of teammates with attention multi-agent ddpg. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1108–1116, 2019.

[25] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 2020.

[26] Afshin OroojlooyJadid and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*, 2019.

[27] D Sai Koti Reddy, Amrita Saha, Srikanth G Tamilselvam, Priyanka Agrawal, and Pankaj Dayama. Risk averse reinforcement learning for mixed multi-agent environments. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2171–2173, 2019.

[28] Walid Saad, Zhu Han, H Vincent Poor, and Tamer Basar. Game-theoretic methods for the smart grid: An overview of microgrid systems, demand-side management, and smart grid communications. *IEEE Signal Processing Magazine*, 29(5):86–105, 2012.

[29] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[30] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[31] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

[32] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

[33] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[34] Ruohan Zhang, Yue Yu, Mahmoud El Chamie, Behçet Açikmese, and Dana H Ballard. Decision-making policies for heterogeneous autonomous multi-agent systems with safety constraints. In *IJCAI*, pages 546–553, 2016.