Domain-Aware Multiagent Reinforcement Learning in Navigation

Ifrah Saeed*, Andrew C. Cullen†, Sarah Erfani† and Tansu Alpcan*
*Department of Electrical and Electronic Engineering, The University of Melbourne, Australia
Email: ifrah.saeed@student.unimelb.edu.au; tansu.alpcan@unimelb.edu.au
† School of Computing and Information Systems, The University of Melbourne, Australia
Email: {andrew.cullen, sarah.erfani}@unimelb.edu.au

***Abstract*—Multiagent reinforcement learning has shown success in guiding the agents' behaviour in systems that have real-world significance. In these frameworks, agents learn how to interact with the environment and other agents while satisfying their objectives. Unfortunately, the level of complexity of real-world problems requires a significant investment of computational resources before multiagent reinforcement learning methods are able to deliver results. However, by incorporating a priori domain knowledge, more computationally-efficient algorithms can be developed. In this paper, for the first time, we present a Domain-Aware Multiagent Actor-Critic (DAMAC) algorithm, which integrates domain knowledge with the centralised learning and decentralised execution multiagent reinforcement learning approach using domain-specific solvers. Our experiments show that our algorithm achieves substantial high reward and reduces the training time by two orders of magnitude as compared to other multiagent reinforcement learning algorithms. This enables the adoption of this powerful framework in more resource-constrained scenarios.**

***Index Terms*—Multiagent reinforcement learning, domain knowledge**

I. Introduction

The ability of **Multiagent Reinforcement Learning (MARL)** to solve real-world problems has been a topic of considerable research interest in areas such as robotics, autonomous vehicle fleets, traffic light control, smart grids, and communication and sensor networks [1]. However, the adoption of MARL, and even single-agent reinforcement learning (RL), has been stymied by their need for significant investments of computational resources, which inherently limits their applicability [2]. To enhance the ability of RL to solve complex and resource-constrained problems of real-world significance, it is important that such RL frameworks are developed that are computationally-efficient. This paper addresses this problem and proposes a MARL algorithm that leverages domain knowledge to improve its sample-efficiency.

Problem domains involving multiple agents add more challenges to learning, as they require balancing the demands of a non-stationary environment and complex credit assignment among multiple agents, while managing the risks of overgeneralisation [3]. One approach for resolving some of these challenges is to use multiagent **Model-free Reinforcement Learning** methods, which translate agents observations to policies without involving the environment. A popular implementation is the centralised training and decentralised execution actor-critic algorithm [4], [5], in which a group of centralised critics (or a single centralised critic) are trained together, with each critic having access to information from all agents. In doing so, the non-stationary nature of the changing policies of the agents can be managed and addressed. However, at the execution time, agents make decisions based upon only the information that they have access to, allowing them to independently interact with the environment.

It must be noted that the model-free approaches induce large number of samples that results in increased computational time and resource usage. Current avenues for improving the computational performance of MARL include **Model-based Reinforcement Learning** [6], in which agents learn and leverage an environment model; **Transfer Learning** [7], in which agents' knowledge is reused to perform similar tasks; and **Imitation Learning** [8], in which agents receive information from humans and experts, and learn a policy by following expert's decisions. However, each of these approaches has been successful in single-agent settings with limited work in multiagent domains. Moreover, transfer and imitation learning techniques require access to information from an oracle that may not be available within the domain context.

Reward Shaping [9] is an effective technique to improve RL computational efficiency by transforming a priori knowledge based on human knowledge and rules into additional numeric rewards. Our approach extends reward shaping in a novel way by using domain-specific solvers to determine the reward shaping function and also incorporating this function while optimising policy. This avoids the need of manually designing the reward function and shows how the incorporation of incomplete but expert available knowledge can help reduce the training time. We demonstrate the value of this approach through a case study of constrained navigation scenarios, in which multiple agents have to collaboratively navigate to reach assigned targets. In such environments, domain knowledge of the environmental layout and any obstacles and bottlenecks therein is often available, making it a perfect example to show the power of using domain knowledge.

By embedding approximate navigational information from a discretised representation of the environment within MARL training, our method aims to improve the computational

This research was funded partially by the Australian Government through the Australian Research Council Discovery Project, DP190102828.

performance. The information is obtained when each agent constructs its own initial domain comprehension under the assumption that there are no other agents present within the domain. This encourages exploration of promising states that leads to lower the computational cost of random exploration. MARL then learns how best to modify this single-agent information, to account both for the presence of other agents, and the differences between the actual continuous-space problem domain and the coarse discretisation. This technique provides adaptability and significantly decreases the overall problem complexity, which in turn allows complex real-world scenarios to be addressed by reinforcement learning.

The constrained navigation environments used for our experiments can act as a digital twin for Industry 4.0 applications including smart factory and other smart manufacturing frameworks [10], [11]. The modeling of such complex environments in not an easy task and using flexible but computationally intensive RL is not practical. Through the use of our novel way of incorporating domain knowledge in RL, we show the ability to significantly decrease the computational cost of training agents in such environments while taking advantage of the flexibility and adaptability of RL. Moreover, this method of utilising domain knowledge is not limited to a single domain and the a priori knowledge from any domain can be incorporated to learn an optimal policy for that domain. We believe that this work will provide an opportunity to explore the use of available knowledge in various domains to devise flexible and computationally-efficient algorithms.

Our paper makes the following contributions:

- We propose Domain-Aware Multiagent Actor-Critic **(DAMAC)**, which incorporates domain knowledge in a novel fashion in the training process of multiagent reinforcement learning. Our approach can be easily incorporated into any centralised training, decentralised execution based MARL algorithm.
- New multiagent navigational scenarios are used to demonstrate the effectiveness of our technique, which by extensive experiments, outperforms other published MARL algorithms in terms of faster convergence and high reward gains.

II. Related Work

One of the common approaches of solving MARL problems is the **centralised learning and decentralised execution actor-critic approach** that includes popular methods such as Multiagent Deep Deterministic Policy Gradient (MADDPG) [4], Counterfactual Multiagent Policy Gradients (COMA) [5], and Multiagent Attention Critic (MAAC) [12]. The main focus of our algorithm is to take advantage of domain knowledge by combining it with the adaptive but inefficient MARL framework to improve its efficiency. In the following, we review recent research to improve MARL computational efficiency.

Model-based RL. Model-based approaches have established research in the single-agent setting presented in the paper [13]. Model-based Multi-Agent Reinforcement Learning with Cooperative Prioritized Sweeping [14] performs sample-efficient multiagent learning by using knowledge about the problem structure in the form of a dynamic decision network to learn the model environment that in turn is used to learn the optimal policy. Model Predictive Control (MPC) Guided Reinforcement Learning Control Scheme [15] combines model-based MPC with RL to improve efficiency. Unlike these model-based approaches, DAMAC does not learn the model or use model-based techniques rather leverages available incomplete knowledge and approximate model to improve the performance of the MARL algorithm.

Transfer learning and imitation learning. The survey [16] covers various techniques of transfer learning in multiagent setting. Another interesting approach is Protean Learning [17] that extends reinforcement learning to adapt to situations that are unknown prior to learning. The application of imitation learning is learning a policy from expert demonstrations but its application is limited on multiagent setting [8]. Moreover, objectives of these learning techniques are to accelerate the learning process but unlike our approach of using available knowledge, they are focused on transferring information from learning agents or experts to other learning agents.

Leveraging domain knowledge. Several approaches currently exist for incorporating domain knowledge within model-free RL including [18], which designs the policy network architecture using hand-designed modular network components for a single-agent RL 'reacher' task to improve the learning efficiency. Alternatively, a role-oriented multiagent reinforcement learning framework (ROMA) [19] combines domain knowledge dependent role assignment with flexible MARL, in order to enable agents with similar responsibilities to share their learning, resulting in improved efficiency. Unlike our approach, these techniques are either focused on single-agent scenarios or use a different way to leverage domain knowledge.

Reward shaping. Reward shaping modifies the reward function in the RL problem using a reward shaping function that incorporates domain knowledge. Potential-based reward shaping (PBRS) [9] is the first and the most popular approach to shape the reward. Important works of multiagent reward shaping include [20], [21], and recent works focus on either learning a reward function or learning to optimise the weight factors of reward functions [22]–[25]. Our method does not use PBRS or learns the reward rather shapes the reward by adding the utility function that is calculated using discretised but single-agent expert domain knowledge that ensures the optimal policy.

III. Problem Statement and Background

In this section, we formally introduce the problem and the necessary notation. We will then provide details of our approach in the next section.

Problem Statement. Current MARL approaches involving deep neural networks suffer from high sample complexity that limits their application on complex real-world problems. Several techniques have been proposed to minimise the computa-

tional time required to reach the maximum achievable reward in MARL—as we discussed in the preceding sections—with those that incorporate domain knowledge showing particular promise. We approach this problem by constructing a single-agent solution using a low-fidelity algorithm, and using information derived from this to bias the MARL training process. Such an approach is advantageous in that it can be readily and rapidly applied to a range of navigation problems in a fashion that yields strong convergence results.

Notation. The multiagent learning setup can be interpreted as Markov or Stochastic Games [26], where a partially observable Markov Game is represented by a collection of states X, a set of actions $U = U_1, U_2, ..., U_N$ for each of N agents, a set of each agent's private observations $O = O_1, O_2, ..., O_N$,and a state transition function $\mathcal{T} : X \times U \times X \rightarrow [0, 1]$ that defines the probability distribution of next possible states and depends on joint action space of all agents. Each agent receives a reward that is given by $r_i : X \times U \times X \rightarrow \mathbb{R}$, which depends on both the state and action spaces of all agents, and is subject to the discount factor $\gamma \in [0, 1]$. The local observation of each agent i, $o_i \in O_i$ contains partial information from the global state $x \in X$. Under these conditions, each agent aims to learn a policy $\pi_i : O_i \rightarrow U_i$ that maximises its total expected return, $\mathbb{E}_{x\sim\mathcal{T},u_1\sim\pi_1,...,u_N\sim\pi_N}\left[\sum_{t=0}^{T}\gamma^t r_{it}(x_t, u_{1t}, ..., u_{Nt})\right]$.

IV. Domain-Aware Multiagent Actor-Critic (DAMAC)

In this section, we present algorithm details and preliminary theoretical analysis of the convergence of this approach. We also provide a case study to explore the benefits of DAMAC.

DAMAC is a MARL algorithm that comprises of off-policy multiagent actor-critic learning and a utility term that incorporates domain knowledge in this trial and error learning. Actor-critic methods are widely used to solve reinforcement learning problems that combine the benefits of policy- and value-based approaches [27]. By incorporating domain knowledge, the reward function is biased towards state-action pairs that have been determined a priori to be favourable. Note that the domain knowledge is approximate and is added to increase the MARL algorithm efficiency and does not directly provide the optimal policy.

During the training phase, agents learn the optimal policy in a centralised manner by taking into account both the information from their experience and available knowledge. After training, the learned policy is executed in a decentralised fashion, in which agents only have access to local observations. During training, the addition of this utility function reduces the time spent in random exploration and helps agents focus on useful state-action pairs that eventually leads to improved sample efficiency and quick convergence to optimal policy learning. This is important to solve complex multiagent problems that either take complete domain knowledge to be solved or take huge computational resources to solve via flexible reinforcement learning techniques.

DAMAC updates the objective of reinforcement learning, $\sum_t \mathbb{E}_\pi[r(x_t, u_t)]$, biased towards state-action pairs suggested by domain knowledge as

$$J(\pi) = \mathbb{E}_\pi\left[r(x_t, u_t) + \beta\lambda(x_t, u_t)\right]. \tag{1}$$

Here $\lambda(x_t, u_t)$ provides the domain-specific information and $\beta \geq 0$ determines the relative emphasis on the domain knowledge utility term with respect to the RL reward. In the case where $\beta = 0$, we recover the RL objective.

Under DAMAC, the critic neural networks of all agents are trained together to minimise the joint regression loss augmented by the utility term through temporal difference learning. For each agent i, there is a corresponding $Q_i^\psi(o, u)$ (where $o = (o_1, ..., o_N)$ and $u = (u_1, ..., u_N)$ for all agents 1 to N) that is the estimate of the true action-value pair, the sum of which is used to measure the total loss. This means that each critic takes observations and actions of all agents. The target Q-function of each agent $Q_i^{\bar{\psi}}(o', u')$ is the exponential moving average of Q-functions, subject to the parameters ψ and $\bar{\psi}$, which respectively represent the critic and target critic. This allows the critic loss to be expressed as

$$\begin{aligned} L_Q(\psi) &= \sum_{i=1}^{N}\mathbb{E}_{(o,u,r,o',\lambda)\sim D}\left[\left(Q_i^\psi(o, u) - y_i\right)^2\right], \text{ where} \\ y_i &= r_i(o_i, u_i) + \beta\lambda_i(o_i, u_i) + \gamma\mathbb{E}_{u'\sim\pi_{\bar{\theta}}(o')}\left[Q_i^{\bar{\psi}}(o', u')\right]. \end{aligned} \tag{2}$$

In this, D represents the experience replay buffer and β is a weighting for the utility term λ.

The policy gradient for each agent to update its policy π_{θ_i} is

$$\begin{aligned} \nabla_{\theta_i} J(\pi_\theta) = \mathbb{E}_{o\sim D, u\sim\pi_\theta}\big[&\nabla_{\theta_i}\log(\pi_{\theta_i}(u_i|o_i)) \\ &(A_i(o, u) + \beta\lambda_i(o_i, u_i))\big], \end{aligned} \tag{3}$$

where θ_i is the policy parameter of agent i. $A_i(o, u)$ is the advantage function that is given by $Q_i^\psi(o, u) - b(o, u_{-i})$ where $-i$ represents all agents except i. The multiagent baseline for our discrete action space is calculated as $b(o, u_{-i}) = \sum_{u_i'}\pi(u_i'|o_i)Q_i(o, (u_i', u_{-i}))$ that overcomes the multiagent credit assignment problem [5] [12]. To accommodate this, the input to the critic Q_i does not include the agent's action u_i so that each critic outputs the Q-value for all actions. Moreover, to avoid overgeneralisation that results in the selection of sub-optimal actions, the sampling of all actions u for policy gradients are done from all agents' current policies [28]. The implementation of DAMAC is given in Algorithm 1.

A. Theoretical Analysis

Following the approach of previous studies [29], we present a preliminary analysis of the off-policy DAMAC algorithm performance through a single-agent domain-aware policy iteration method. In this analysis, we assume that accurate domain knowledge is available. The general policy iteration algorithm alternates between policy evaluation and policy improvement. To enable convergence guarantees, we first show that policy iteration under the new objective (Equation 1), converges to an optimal policy in the tabular setting.

Algorithm 1 DAMAC algorithm

```
Initialise critic and policy parameters ψ, ψ̄, θ, θ̄.
for episode = 1 to number-of-episodes do
    Calculate λ_i given initial o_i and domain knowledge.
    for t = 1 to total-episode-steps do
        Retrieve λ_i based on current o_i of each agent i.
        Apply action u_i ∼ π_i(.|o_i) and update
        (o_i, u_i, r_i, o'_i, λ_i ) in the replay buffer D.
        for j = 1 to number-of-critic-updates do
            Sample m samples, (o^m, u^m, r^m, o'^m, λ^m) ∼ D
            for all agents.
            Update critic using Equation 2.
        end for
        for j = 1 ... number-of-policy-updates do
            Sample s samples (o^s) ∼ D.
            Update policies using Equation 3.
        end for
        Update target critic and policy parameters:
        ψ̄ = τψ̄ + (1 − τ)ψ and θ̄ = τθ̄ + (1 − τ)θ
    end for
end for
```

Proposition 1 (Policy Evaluation). To compute the Q-value of a policy π, Q_π, with respect to the updated RL objective in Equation 1, the Q-value is updated iteratively until convergence as $\lim_{k\to\infty}(\mathcal{T}^\pi)^k Q = Q_\pi$, with Q as the initial value of the policy and the Bellman backup operator is

$$\mathcal{T}^\pi Q(x_t, u_t) \doteq r(x_t, u_t) + \beta\lambda(x_t, u_t) + \gamma\mathbb{E}_{x_{t+1}\sim T}\left[V(x_t)\right],$$

where $V(x_t) = \mathbb{E}_{u_{t+1}\sim\pi} Q(x_{t+1}, u_{t+1})$.

Proof. The $\beta\lambda(x_t, u_t)$ term updates the reward $r(x_t, u_t)$ as

$$r'(x_t, u_t) = r(x_t, u_t) + \beta\lambda(x_t, u_t). \tag{4}$$

If $\beta\lambda(x_t, u_t)$ is strictly positive, then the standard convergence analysis results for policy evaluation [30] still holds.

Proposition 2 (Policy Improvement). For policies π and π', if $Q_{\pi'}(x_t, u_t) \geq Q_\pi(x_t, u_t)$ for all $(x_t, u_t) \in X \times U$ then π' is better than π.

Proof. The Q-value of a policy π for DAMAC on applying Equation 4 is given by

$$Q_\pi(x_t, u_t) \doteq r'(x_t, u_t) + \gamma\mathbb{E}_{x_{t+1}\sim T}\left[V_\pi(x_{t+1})\right]. \tag{5}$$

Moreover, the new policy π' is chosen greedily by selecting $\pi'(x_t) = \text{argmax}_{u_t} Q_\pi(x_t, u_t)$ where Q_π is more for state-action pairs with higher $\beta\lambda(x_t, u_t)$. Hence, $Q_\pi(x_t, \pi'(x_t)) \geq V_\pi(x_t)$ and by repeated applications of Equation 5, it must also be true that

$$\begin{aligned} Q_\pi(x_t, u_t) &\leq r'(x_t, u_t) + \\ &\quad \gamma\mathbb{E}_{x_{t+1}\sim T}\left[\mathbb{E}_{u_{t+1}\sim\pi'}\left[Q_\pi(x_{t+1}, u_{t+1})\right]\right] \\ &\quad \leq Q_{\pi'}(x_t, u_t), \end{aligned}$$

where convergence to Q'_π is proved in Proposition 1. Note that $\beta\lambda(x_t, u_t)$ represents expert domain knowledge so this non-potential reward shaping keeps the optimal policy invariant [9]. However, in DAMAC, we use single-agent domain knowledge in multiagent domain so the policy improvement is dependent on the weighting factor β. This dependence has been shown by empirical analysis in Section V-B.

Policy Iteration. Alternating between policy evaluation and policy improvement yields policy iteration and it converges to optimal policy π^* such that $Q_{\pi^*}(x_t, u_t) \geq Q_\pi(x_t, u_t)$, for all $\pi \neq \pi^*$ and for all $(x_t, u_t) \in X \times U$ [30].

To apply the policy iteration algorithm on complex domains, we approximate it by using neural networks for both Q-function and the policy instead of finding the optimal solution in its exact form. Therefore, we no longer run evaluation and improvement steps to convergence rather we alternate between optimising both networks with stochastic gradient descent.

B. Case Study: Constrained Navigational Scenarios

DAMAC is most useful in complex scenarios where we require both the flexibility obtained from RL algorithms to achieve goals, and less computational time and resources to train these flexible algorithms. To explore the relative advantages of DAMAC, we consider two novel scenarios, that capture some of the difficulties in performing complex constrained navigation. These problems are formulated as multiagent navigation tasks, where each agent has to reach a predetermined target while avoiding both domain obstacles and other agents. The latter of these goals is achieved through the development of emergent collaborative behaviour using MARL.

The domain knowledge is captured in the utility function λ, given by Equation 1 that parameterises a single-agent shortest path routing algorithm. To calculate λ, a single-agent A* path finding algorithm is employed to determine the optimal route for each agent to its destination, based on a coarsely discretised representation of the continuous-spaced navigation environment. This makes λ dependent on the state for this case study. As this utility is constructed for a single-agent A* path that ignores the presence of other agents on a discretised grid, it can be trivially precomputed with minimal cost, as described in Algorithm 2. Through the addition of this approximate utility term in MARL training, the complexity of solving the navigation scenario is reduced, as it incentivises agent's observations that lead to more reward by making each agent follow the route suggested by the A* algorithm.

As the path returned by A* is one of the many possible shortest paths for a single-agent on a coarsely discretised grid, the optimal path for multiple agents will likely differ from the path suggested by λ. As such, the incorporation of domain knowledge acts as a catalyst to increase the MARL algorithm efficiency and does not directly provide the optimal path. Moreover, while the calculation of the utility function has been presented in terms of A*, the process for incorporating a priori knowledge can leverage any extant path search function, including but not limited to Dijkstra's algorithm, breadth first search, and hierarchical algorithms [31]. We chose A* in our case for its simple implementation and wide-spread usage in navigational environments and games. Moreover, DAMAC can

Algorithm 2 λ for constrained navigation

Discretise continuous environment into an $m \times n$ grid.
for each agent **do**
  Translate agent and target positions to their respective locations upon the discretised grid.
  Calculate the shortest path and the cost to reach all cells of the discrete grid with A*.
  Add the largest possible cost within the environment (max) to the shortest path cost.
end for
Normalise by dividing all costs by $2 \times max$ within the environment to keep the value $(0, 1]$.
The normalised cost of the shortest path is λ.

potentially be used for tasks beyond navigation where domain knowledge is available and a utility function can capture this a priori knowledge.

V. Experiments

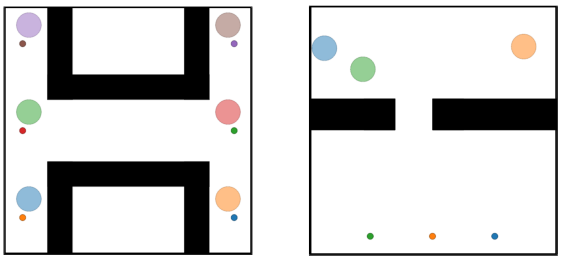
In this section, we provide the details of constrained navigational environments used to test DAMAC and our experimental setup. We also compare the efficiency of DAMAC with other MARL methods on our environments and analyse the results.

A. Environments and Setup

Simulated Environments. We use Multiagent Particle Environment (MPE) [32] to construct two novel environments, Crossing Corridor and Bottleneck that represent the complex constrained navigation. MPE is frequently used in the literature for testing the multiagent RL algorithms and many new scenarios are built on top of the existing ones that confirms its flexibility [4], [12].

Crossing Corridor. In this scenario, there are a total of 6 agents, each with a corresponding target, as shown in Fig. 1(a). The role of each agent is to navigate its way to reach its respective goal in the shortest amount of time without colliding with other agents. The initial positions of agents are randomised to either the left or right side of the environment. To make the scenario challenging, the agents on the left-hand side have their corresponding targets on the right-hand side and vice versa. Agents are penalized on colliding with one another and based on their distances from their destinations. All agents receive a global reward when any agent reaches and stays at its target. The total reward attained by agents is the sum of all rewards and penalties that agents get during the length of each episode (epoch). In this scenario, since agents and their destinations are not always in line of sight so agents take a considerable amount of time to learn to reach their respective goals by randomly exploring the environment.

Bottleneck. Fig. 1(b) shows the Bottleneck environment that involves 3 agents on the upper portion and their corresponding 3 destinations in the lower portion under the wall. Only a single-agent can pass through the wall opening at a time and hence the name, Bottleneck. All agents have to learn to collaboratively reach their respective targets making their way through the bottleneck without colliding with one another. Agents' positions are initialised to the upper portion of the environment and are different at the start of each episode (epoch). The reward function in this scenario is same as that in Crossing Corridor scenario. However, this scenario tests the algorithm ability to train all agents to not only find their paths to reach their respective destinations but to see how agents are prioritised to pass through the bottleneck without colliding with one another.



(a) Crossing Corridor (b) Bottleneck

Fig. 1: Scenario environments. Agents and their targets are respectively represented by large and small same-coloured circles.

Experimental Setup. For our experiments, we use the replay buffer size of 10^6 that is filled with agents' experience $(o_t, u_t, r_i, o_{t+1}, \lambda_t)$ in 8 parallel rollouts at every timestep for each independent simulation. We use batchsize of 1024 and update the parameters of our policy, critic, and target networks 4 times after every 100 steps across all rollouts. We use the Adam optimiser to update both policy and critic networks with the learning rate set to 0.001. To update both policy and critic target networks, τ is also set to 0.001. We set the value of the discount factor γ as 0.99. All policy and critic networks use 2 hidden layered neural network with each hidden dimension of 128 and leaky rectified linear units as the nonlinearity.

B. Results and Discussion

Baselines. We compare our approach with the following widely-used centralised learning and decentralised execution MARL techniques: Multiagent Deep Deterministic Policy Gradient (MAADDPG) [4], Counterfactual Multiagent Policy Gradients (COMA) [5], and Multiagent Attention Actor-Critic (MAAC) [12] on our constrained navigational environments. We also use an ablated version of DAMAC with $\beta = 0$ in which domain knowledge has not been incorporated and agents learn to find the optimal solution by trial and error using only MARL framework. We include this case in graphs as a visual reference point to highlight the importance of including available knowledge within the DAMAC framework. We discuss $\beta = 0$ results in detail when analysing the effect of utility weighting factor. Based on performance, we tune the hyperparameters of each algorithm that end up being same as that of DAMAC. We use the episode (epoch) length of 400 and 200 for Crossing Corridor and Bottleneck respectively. Moreover, we use $\beta = 1$ for DAMAC unless otherwise stated. Please note that given the nature of our approach, it can be used with other MARL algorithms to improve their performance.

Analysis. The reward function plotted in Fig. 2 is the sum of all accumulated rewards across the entire episode aver-

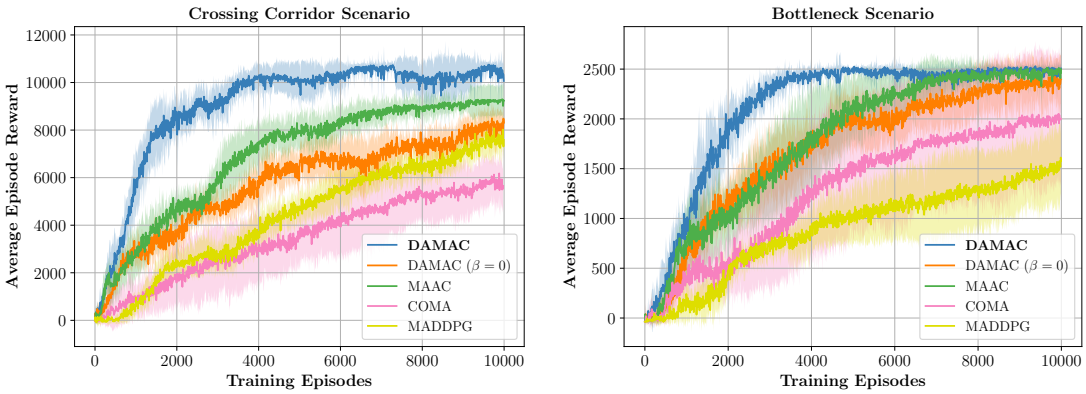


Fig. 2: Reward achieved by methods, DAMAC (ours), DAMAC without domain knowledge ($\beta = 0$), MAAC [12], MADDPG [4] on Crossing Corridor and Bottleneck. Average (solid colour) and standard deviation (shaded region) across six independent simulations following established practice in MARL.

TABLE I: Training efficiency of multiple MARL approaches

Crossing Corridor	DAMAC	MAAC	COMA	MADDPG
Highest reward	10752 ± 147	9296 ± 666	6178 ± 860	8072 ± 758
Epochs for 95% reward	3760	6608	9408	9288
Bottleneck				
Highest reward	2529 ± 20	2518 ± 53	2051 ± 658	1612 ± 358
Epochs for 95% reward	3168	6536	8280	9272

aged over six independent simulations following established practices for MARL. As evident from the figure, DAMAC outperforms the reference techniques with respect to both the rate of convergence and the highest achieved reward since the reward shaping minimises the time that agents need to explore and learn about navigating the environment. The reference algorithms take significantly longer to explore enough of the domain to begin refining their own behaviours, resulting in the slower or no convergence behaviour within the given training time.

From Fig. 2 and Table I, MAAC reaches its highest cumulative reward value of 9296 in **Crossing Corridor** in 9952 episodes whereas DAMAC achieves the same reward in 2480 episodes, which is only **24.9%** of the episodes taken by MAAC. Furthermore, DAMAC is able to achieve a higher reward of 10752, which is a **15.71%** improvement over MAAC. DAMAC also attains **26.6%**, **74%**, and **33.2%** more reward as compared to DAMAC without domain knowledge, COMA, and MADDPG respectively that achieve the maximum reward of 8495, 6178, and 8072 in $10,000$ episodes respectively. In **Bottleneck**, MAAC reaches its highest cumulative reward of 2518 in 9032 epochs whereas DAMAC attains this reward in 4760 episodes, which is only approximately half of the episodes as compared to MAAC. DAMAC without domain knowledge, COMA, and MADDPG attain their highest rewards of 2424, 2051, and 1612 respectively as compared to the 2529 convergence reward of DAMAC. This shows that DAMAC exhibits the maximum reward gain of **4.3%**, **23.3%**, and **56.9%** over DAMAC without domain knowledge, COMA, and MADDPG in Bottleneck in 10k epochs. Since DAMAC with $\beta = 0$ is another MARL algorithm that does not take advantage of domain knowledge, its performance is closer to MAAC in both scenarios with MAAC slightly better as MAAC takes advantage of attention and maximum entropy.

In the experiments, the average maximum reward fluctuates once the algorithm reaches the convergence point as it depends on the initial positions of agents. For example, if the positions of agents are closer to their destinations at the start of episodes, the reward is higher in these episodes. This also explains the variance of MARL methods evident in Bottleneck.

Effect of Episode Length. Given the complexity of these constrained environments, we choose the episode length of 400 and 200 for Crossing Corridor and Bottleneck respectively. To test the sensitivity of the calculated results to the episode length, we increase the episode length for Crossing Corridor to 500 and repeat our experiments. With the increased episode length, agents have more time to explore the environment in a single episode rather than settling to a sub-optimal policy because of the lack of exploration. However, this exploration is at the expense of 25% more computation time each episode, the reduction of which is our primary goal in this paper. Table II shows the comparison of DAMAC with other MARL approaches in Crossing Corridor when its training episode length is increased to 500 from 400, that was reported in Table I. The numbers are averaged over six independent runs.

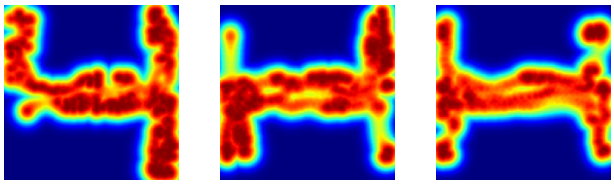
From Table II, both DAMAC and MAAC attain convergence to the highest reward but DAMAC takes only half of the episodes that MAAC requires to reach convergence. Therefore, while increasing the episode length does improve the results of all techniques, DAMAC still significantly outperforms all other techniques in terms of higher reward gain and faster convergence. Moreover, increase in the episode length introduces a proportional increase to the overall computational cost. Note that the highest reward for an environment is different for

TABLE II: Comparison of DAMAC with reference MARL methods with episode length **= 500** for Crossing Corridor

	DAMAC	MAAC	COMA	MADDPG
Highest reward	13797 ± 93	13218 ± 868	10528 ± 2297	9209 ± 242
Epochs for 95% reward	3512	5520	8528	8464

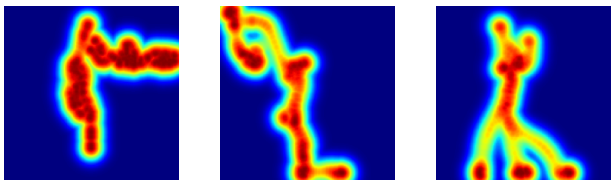
different episode lengths because the reward is accumulated across the whole epoch size.

Learned Behaviours. We illustrate the evolution of agents' learned behaviour with DAMAC using heat maps in Figures 3 and 4. For each environment, these heat maps exhibit the agents' evolution from initial exploration (a); to incorporating domain knowledge to produce a sub-optimal route to the destination (b); and finally refining the path to produce a more optimal path that minimises collisions to reach the destination in fewer steps (c). This convergence process is clear through the coalescence of the heat map trajectories in initial episodes to a more condensed set of positions within the environment in later episodes when the optimal policy is learned.



(a) Episode 100 (b) Episode 2000 (c) Episode 4000

Fig. 3: Agents' heatmaps for Crossing Corridor after different levels of learning.



(a) Episode 100 (b) Episode 1000 (c) Episode 3000

Fig. 4: Agents' heatmaps for Bottleneck after different levels of learning.

Ablation Analysis and Effect of Utility Weighting Factor. As already mentioned, the utility value is normalised to keep its value in the range (0, 1] so that incorporating the utility calculated for other environments is straight-forward. In this section, we note the influence of utility weighting factor β, by analysing algorithmic performance in terms of its convergence and the highest achieved reward. Domain knowledge is not incorporated with MARL when $\beta = 0$ and it constitutes only MARL framework. With increased value of β, the focus is more on approximate domain knowledge that is determined for a single-agent in a discretised environment and does not take into account other agents in the environment.

Table III summarises the highest reward that DAMAC achieves with different values of β and the number of training episodes at which 90% of the overall highest reward across all values of β is achieved. As the value of β increases, the highest reward achieved by the algorithm decreases and even results in sub-optimal policies where not all agents reach their targets. Agents are inclined to take the path returned by the A* algorithm that is not optimal in the multiagent setting with this increase in β. Therefore, because of the walls, bottlenecks, and other agents, collisions occur that result in the reduction of the total received reward. Moreover, the utility term is similarly high for locations nearby the goal as a consequence of the discretisation of the continuous space. This results in agents resorting to non-optimal policies where they don't explore further to reach their actual targets once they reach the locations near their goals.

In these experiments with varying β values, if the initial positions of agents are such that there are less collisions among agents, and agents and goals are in line-of-sight, the achieved total reward is more for those runs. Hence, for higher values of β, there is more divergent behaviour in independent runs that can be seen in the table III with $\beta \geq 3$. The case, $\beta = 0$ acts poorly in Crossing Corridor because of the complexity of the environment but manages to achieve the highest reward in Bottleneck with a very slow convergence rate. This shows that as the complexity of the environment increases, incorporating domain knowledge enhances convergence. Moreover, it is a coincidence that $\beta = 1$ comes out to be the nominal β value in both cases as it cannot be generalised to all environments. Therefore, the balance between MARL and available knowledge is the key to get the optimal policy in the minimum number of training epochs in navigational environments.

Importance of Domain Knowledge in MARL. Through experiments and ablation analysis of DAMAC, we elucidate the value of domain knowledge in MARL. Our incomplete but expert domain knowledge incentivises one of the multiple possible single-agent paths returned by the low resolution A* algorithm, with a positive reward. This information significantly improves the achieved rewards and convergence rate in our constructed experiments, relative to more computationally complex MARL approaches. This performance occurs even in environments where collision events are highly probable along the domain knowledge provided shortest path. This demonstrates that agents do not just learn to follow the domain knowledge, but leverage it alongside their own experience to better navigate around both the environment and the other agents contained therein.

VI. Conclusion

In this work, we have presented an algorithmic approach of using domain-specific solvers to incorporate domain knowledge in a multiagent reinforcement learning framework. We propagate the domain knowledge through the critic networks loss function, which in turn influences the overall policy. Using a case study of two constrained navigational environments, we demonstrate that the computational efficiency of reinforcement learning methods is considerably improved by utilising the raw form of available knowledge. Our experiments show that our algorithm, DAMAC, takes half and even less number

TABLE III: Effect of β on DAMAC Learning

Crossing Corridor	$\beta = 0$	$\beta = 1$	$\beta = 2$	$\beta = 3$	$\beta = 4$	$\beta = 5$	$\beta = 6$
Highest reward	8495 ± 683	10752 ± 147	10581 ± 244	10022 ± 816	8965 ± 1083	9988 ± 914	9327 ± 718
Epochs for 90% h_{rew}	Not achieved	3280	4448	6536	Not achieved	9184	Not achieved
Bottleneck							
Highest reward	2424 ± 149	2529 ± 205	2510 ± 25	2262 ± 417	1914 ± 358	2087 ± 444	2460 ± 25
Epochs for 90% h_{rew}	7136	2744	4424	Not achieved	Not achieved	Not achieved	3520

Highest reward, and epochs that correspond to the point of reaching 90% of the highest reward (across all β, with h_{rew} for 10752 and 2529 for Crossing Corridor and Bottleneck respectively), using the average of 6 independent simulations.

of training episodes to converge than those required by the existing MARL approaches in two constrained environments. Moreover, it achieves a substantial higher reward as compared to other methods. We have also found out that a balance between the approximate domain knowledge and reinforcement learning to train agents is required to obtain the optimal policy as weighting the approximate knowledge more leads to lower rewards. In summary, our approach lays the foundations for intelligently incorporating a priori knowledge for complex, multiagent reinforcement learning tasks. In doing so, MARL may be flexibly applied to practical scenarios in an Industry 4.0 context, including smart factories and other automation frameworks.

REFERENCES

[1] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *arXiv preprint arXiv:1911.10635*, 2019.

[2] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *arXiv preprint arXiv:2007.05558*, 2020.

[3] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, 2019.

[4] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[5] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, pp. 2974–2982.

[6] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based reinforcement learning: A survey," *arXiv preprint arXiv:2006.16712*, 2020.

[7] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research*, vol. 10, no. 7, pp. 1633–1685, 2009.

[8] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[9] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of International Conference on Machine Learning*, vol. 99, 1999, pp. 278–287.

[10] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination," *Computer Networks*, vol. 101, pp. 158–168, 2016.

[11] B. Klenz, "How to use deep learning with your internet of things (IoT) digital twin," *Proceedings of the SAS Global Forum 2019 Conference*, 2019.

[12] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, vol. 97, 2019, pp. 2961–2970.

[13] E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking model-based reinforcement learning," *arXiv preprint arXiv:1907.02057*, 2019.

[14] E. Bargiacchi, T. Verstraeten, D. M. Roijers, and A. Nowé, "Model-based multi-agent reinforcement learning with cooperative prioritized sweeping," *arXiv preprint arXiv:2001.07527*, 2020.

[15] H. Xie, X. Xu, Y. Li, W. Hong, and J. Shi, "Model predictive control guided reinforcement learning control scheme," in *Proceedings of the International Joint Conference on Neural Networks*, 2020, pp. 1–8.

[16] F. L. Da Silva and A. H. R. Costa, "A survey on transfer learning for multiagent reinforcement learning systems," *Journal of Artificial Intelligence Research*, vol. 64, pp. 645–703, 2019.

[17] I. Bonnici, A. Gouaïch, and F. Michel, "Input addition and deletion in reinforcement: Towards learning with structural changes," in *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 177–185.

[18] R. Ramamurthy, C. Bauckhage, R. Sifa, J. Schücker, and S. Wrobel, "Leveraging domain knowledge for reinforcement learning using MMC architectures," in *Proceedings of the International Conference on Artificial Neural Networks*, 2019, pp. 595–607.

[19] T. Wang, H. Dong, V. Lesser, and C. Zhang, "ROMA: Multi-agent reinforcement learning with emergent roles," in *Proceedings of the International Conference on Machine Learning*, 2020, pp. 9876–9886.

[20] S. Devlin and D. Kudenko, "Theoretical considerations of potential-based reward shaping for multi-agent systems," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2011, pp. 225–232.

[21] F.-Y. Sun, Y.-Y. Chang, Y.-H. Wu, and S.-D. Lin, "Designing non-greedy reinforcement learning agents with diminishing reward shaping," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018, pp. 297–302.

[22] Z. Zheng, J. Oh, and S. Singh, "On learning intrinsic rewards for policy gradient methods," in *Proceedings of Advances in Neural Information Processing Systems*, 2018, pp. 4644–4654.

[23] Z.-Y. Fu, D.-C. Zhan, X.-C. Li, and Y.-X. Lu, "Automatic successive reinforcement learning with multiple auxiliary rewards." in *Proceedings of International Joint Conferences on Artificial Intelligence*, 2019, pp. 2336–2342.

[24] Y. Sun and F. Huang, "Can agents learn by analogy? an inferable model for pac reinforcement learning," in *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1332–1340.

[25] Z. Zheng, J. Oh, M. Hessel, Z. Xu, M. Kroiss, H. Van Hasselt, D. Silver, and S. Singh, "What can learned intrinsic rewards capture?" in *Proceedings of International Conference on Machine Learning*, 2020, pp. 11 436–11 446.

[26] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the International Conference of Machine Learning*, 1994, pp. 157–163.

[27] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proceedings of Advances in neural information processing systems*, 2000, pp. 1008–1014.

[28] E. Wei, D. Wicke, D. Freelan, and S. Luke, "Multiagent soft q-learning," in *Proceedings of AAAI Spring Symposium Series*. AAAI Press, 2018.

[29] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 1861–1870.

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

[31] D. Delling, P. Sanders, D. Schultes, and D. Wagner, "Engineering route planning algorithms," in *Proceedings of Algorithmics of large and complex networks*, 2009, pp. 117–139.

[32] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 1495–1502.