Deep Implicit Coordination Graphs for Multi-agent Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Multi-agent reinforcement learning (MARL) requires coordination to efficiently solve certain tasks. Fully centralized control is often infeasible in such domains due to the size of joint action spaces. Coordination graph based formalization allows reasoning about the joint action based on the structure of interactions. However, they often require domain expertise in their design. This paper introduces the *deep implicit coordination graph* (DICG) architecture for such scenarios. DICG consists of a module for inferring the dynamic coordination graph structure which is then used by a graph neural network based module to learn to implicitly reason about the joint actions or values. DICG allows learning the tradeoff between full centralization and decentralization via standard actor-critic methods to significantly improve coordination for domains with large number of agents. We apply DICG to both centralized-training-centralized-execution and centralized-training-decentralized-execution regimes. We demonstrate that DICG solves the *relative overgeneralization* pathology in predatory-prey tasks as well as outperforms various MARL baselines on the challenging StarCraft II Multi-agent Challenge (SMAC) and traffic junction environments.

1 Introduction

Effective multi-agent reinforcement learning (MARL) in fully cooperative environments often requires coordination between agents on a team. One simple approach for achieving coordination is to reduce the problem to a single agent problem where the action space is the joint action space of all agents. Unfortunately, this joint action space grows exponentially with the number of agents, making it intractable for many domains of interest. To avoid this problem, a common strategy is to *decentralize* or factorize the decision policy or value function for each agent [1–3]. Each agent selects actions to maximize its corresponding utility function, with the end goal of maximizing the joint value function. However, such decentralization can be suboptimal [4]. The optimal policy is often not learnable in such a context due to a game-theoretic pathology referred to as *relative overgeneralization* [5], where the agent's reward gets confounded by penalties from random exploratory actions of other collaborating agents.

Guestrin *et al.* [6] introduced the framework of *coordination graph* (CG) to reason about joint value estimates from a factored representation to significantly improve computational tractability at the expense of optimality. Compared to function decomposition schemes like Value Decomposition Networks [7], QMIX [8], and parameter sharing in decentralized policy optimization [2], the CG framework allows explicit modeling of the locality of interactions and formal reasoning about joint actions given the coordination graph structure. Kok and Vlassis [9] applied these ideas in the context of tabular reinforcement learning. The approach was later extended to function approximation with neural networks by Böhmer *et al.* [10]. Most of these approaches assume a domain dependent static coordination graph is given. However, for a wide range of problems, this coordination graph structure is dynamic and state dependent. Domain heuristics like adding a graph edge with neighboring agents based on some distance metric are sometimes used [11]. The approach of Kok *et al.* [12] attempts to learn such structure, but it is limited to tabular settings with domain heuristics. We hypothesize that methods that learn the appropriate dynamic coordination graph to inform the selection of joint actions can help address coordination issues in MARL.

We propose the *Deep Implicit Coordination Graph* (DICG) module for multi-agent deep RL. It uses a self-attention network to determine an implicit coordination graph structure which is then used for agent information integration through a graph convolutional network (GCN). The intuition behind this architecture design is to make both the coordination graph structure and action inference over its edges differentiable so that the entire DICG module can be used inside either the actor or the critic and trained end-to-end through standard policy optimization methods. Since the module is trained to optimize the joint reward, the GCN submodule learns to implicitly reason about joint actions/values based on the structure of interaction inferred by the attention submodule.

We compare DICG's performance with that of fully centralized and decentralized MARL methods in a challenging domain involving predator-prey tasks that require strong coordination. We also study performance on the StarCraft II Multi-Agent Challenge (SMAC) [13] and the traffic junction environment [3]. DICG learns the relevant dynamic coordination graph structure, allowing it to make an appropriate trade-off between centralized and decentralized methods to outperform standard actor-critic benchmarks on these tasks.

2 Background and Related Work

We formalize the problem as a Dec-POMDP [14] $\langle \mathcal{I}, \mathcal{S}, \{\mathcal{A}^i\}_{i=1}^n, \mathcal{T}, \mathcal{Z}, R, \mathcal{O}, \gamma \rangle$, where $\mathcal{I} = \{1, \dots, n\}$ is the set of agents, $\mathcal{S}$ is the global state space, $\mathcal{A}^i$ is the action space of the ith agent, and $\mathcal{Z}$ is the observation space for an agent. The transition function defining the next state distribution is given by $\mathcal{T} : \mathcal{S} \times \prod_i \mathcal{A}^i \times \mathcal{S} \rightarrow [0, 1]$. The reward function is $R : \mathcal{S} \times \prod_i \mathcal{A}^i \rightarrow \mathbb{R}$, and the discount factor is $\gamma \in [0, 1)$. The observation model defining the observation distribution from the current state is $\mathcal{O} : \mathcal{S} \times \mathcal{Z} \rightarrow [0, 1]$. Each agent i has a stochastic policy π^i conditioned on its observations o_i or action-observation history $\tau^i \in (\mathcal{Z} \times \mathcal{A}^i)$. The discounted return is $G_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$, where r_t is the joint reward at step t. The joint policy π induces a value function $V^\pi(s_t) = \mathbb{E}[G_t \mid s_t]$ and an action-value function $Q^\pi(s_t, \mathbf{a}_t) = \mathbb{E}[G_t \mid s_t, \mathbf{a}_t]$, where $\mathbf{a}_t$ is the joint action. The advantage function is then $A^\pi(s_t, \mathbf{a}_t) = Q^\pi(s_t, \mathbf{a}_t) - V^\pi(s_t)$.

2.1 Policy Optimization

We use policy optimization to maximize the expected discounted return. Given policy π_θ parameterized by θ, the surrogate policy optimization objective is [15]:

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] \tag{1}$$

where $\hat{A}_t$ is the advantage function estimator [16] at time step t and the expectation $\hat{\mathbb{E}}_t[\dots]$ indicates the empirical average over a finite batch of samples. In practice, we use the clipped PPO objective [15] to limit the step size for stable updates. For the policy π_θ, we can either condition on states using a feed-forward network like multi-layer perceptron (MLP), or condition on the full history using a recurrent neural network such as an LSTM [17] or GRU [18]. In the context of centralized training but decentralized execution, a common strategy is to share the policy parameters between agents that are homogeneous [2, 10]. With shared rewards, COMA [19] critic can be useful for better credit assignment and can be easily combined with our proposed approach. However, these approaches do not model the coordination structure. Wei *et al.* [20] investigate relative overgeneralization in continuous action multi-agent tasks and show improvement over MADDPG [21]. OroojlooyJadid and Hajinezhad [22] provide a general overview of cooperative deep MARL.

2.2 Coordination Graphs

For several multi-agent domains, the outcome of an agent's action often depends only on a subset of other agents in the domain. This *locality of interaction* can be encoded in the form of a coordination graph (CG) [6]. A CG is often represented as an undirected graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$ and contains a vertex $v_i \in \mathcal{V}$ for each agent i and a set of undirected edges $\{i, j\} \in \mathcal{E}$ between vertices v_i and v_j. A CG induces a factorization of an action-value function into *utility functions* f^i and *payoff functions* f^{ij}:

$$q^{\text{CG}}(s_t, a) = \sum_{v^i \in \mathcal{V}} f^i(a^i \mid s_t) + \sum_{\{i,j\} \in \mathcal{E}} f^{ij}(a^i, a^j \mid s_t) \tag{2}$$

Guestrin *et al.* [6] and Vlassis *et al.* [23] draw on the connections with maximum-aposteriori (MAP) estimation techniques in probabilistic inference to compute the joint action from such factorizations; resulting into algorithms like Variable Elimination and Max-Plus. Kok and Vlassis [9] explored their use in the context of tabular MARL. Deep Coordination Graphs (DCG) [10] extended these ideas of factoring the joint value function of all agents according to a static coordination graph into payoffs between pairs of agents to deep MARL. They did so by estimating the payoff functions using neural networks and using message passing based on Max-Plus [23] along the coordination graph to maximize the value function, allowing training of the value function end-to-end with Q-learning.

In this work, however, we forgo explicitly computing the joint action through inference over factored representation with a given coordination graph. Instead, we use attention to learn the appropriate agent observation-dependent coordination graph structure with soft edge weights and then use message passing in a graph neural network to compute appropriate values or actions for the agents, such that the computation graph remains differentiable.

2.3 Self-attention

Self-attention mechanism [24] emerged from the natural language processing community. It is used to relate different positions of a single sequence. The difference between self-attention and standard attention is that self-attention uses a single sequence as both its source and target sequence. It has been shown useful in image caption generation [25, 26] and machine reading [24, 27].

The attention mechanism has also been adopted recently for MARL. The relations between a group of agents can be learned through attention. Iqbal and Sha [28] use attention to extract relevant information of each agent from the other agents. Jiang and Lu [29] use self-attention to learn when to communicate with neighboring agents. Wright and Horowitz [30] use self-attention on the policy level to differentiate different types of connections between agents. Jiang *et al.* [11] use multi-head dot product attention to compute interactions between neighbouring agents for the purpose of enlarging agents' receptive fields and extracting latent features of observations. We use self-attention to learn the attention weights between agents, and use the attention weights to form a "soft"-edged coordination graph instead of edges with binary weights.

2.4 Graph Neural Networks

Several frameworks have been proposed to extract locally connected features from arbitrary graphs [31, 32]. Given a graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$, a graph convolutional network (GCN) takes as input the feature matrix that summarizes the attributes of each node $v_i \in \mathcal{V}$ and outputs a node-level feature matrix. This is similar to how a convolution operation across local regions of the input produces feature maps in CNNs. MAGnet learns relation weights through a loss function based on heuristic rules in the form of a relevance graph [33]. Deep relational RL embeds multi-head dot-product attention as relational block into graph neural networks to learn pairwise interaction representation of a set of entities in the agent's state [34]. Recently, Liu *et al.* [35] combined a two-stage attention network with a graph neural network for communication between the agents to achieve state-of-the-art performance on the traffic junction domain with curriculum training [36].

3 Approach

Instead of the standard approach of learning the binary weights of the edges in a coordination graph, we use self-attention to learn the relation between agents and use the attention weights as soft edges of a coordination graph. These soft edges form an implicit coordination graph representing elements of its adjacency matrix, $M \in \mathbb{R}^{n \times n}_{>0}$. We use self-attention to avoid building coordination graphs using hard-coded or domain-specific heuristics so that our approach is applicable to more abstract multi-agent domains. Moreover, maintaining differentiability is difficult with binary connections. We

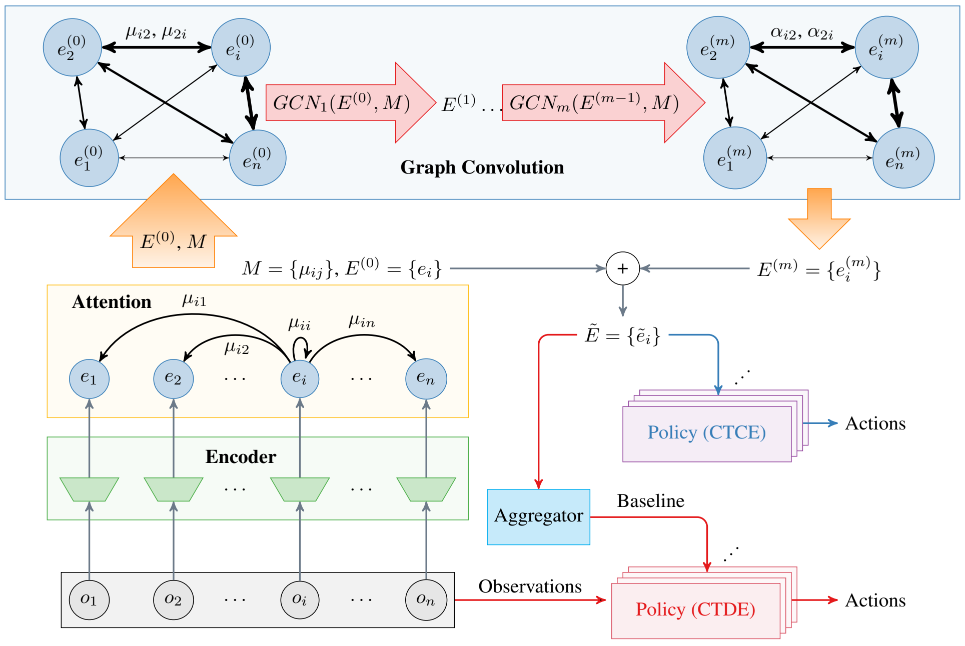

Figure 1: Network architecture of DICG. It can be used for either a centralized-training-centralized-execution (CTCE) approach or as a centralized-training-decentralized-execution (CTDE) approach. The blue arrows indicate the CTCE approach. The DICG module serves as a joint observation encoder. We use the integrated observations $\tilde{E}$ to directly obtain actions for agents through a parameter sharing policy. The baselines in CTCE are estimated by a concatenation of raw observations. The red arrows indicate the CTDE approach. We pass the integrated observations $\tilde{E}$ through an aggregator network to estimate a centralized baseline. We then use the baseline to compute the advantage to guide policy optimization.

use attention to implicitly represent the edge weights as the strength of the connection between agents to obtain the graphs's adjacency matrix. We then apply graph convolution [31] with this adjacency matrix to integrate information across agents. We use graph convolution because it is an efficient and differentiable way to pass information along the graph. With the integrated information, we can either use it as observation embeddings to directly obtain actions or use it to estimate baselines for advantage estimation during policy optimization. In summary, the DICG module consists of an encoder, an attention module, and a graph convolution module with the architecture outlined in Fig. 1.

In detail, we first pass n observations $\{o_i\}_{i=1}^n$ of the n agents through a parameter sharing encoder parameterized by θ_e. The encoder outputs n embedding vectors $\{e_i\}_{i=1}^n$, each with size d:

$$e_i = \text{Encoder}(o_i; \theta_e), \text{ for } i = 1, \dots, n. \tag{3}$$

We then compute the attention weights from agent i to j using these embeddings as:

$$\mu_{ij} = \frac{\exp(\text{Attention}(e_i, e_j, W_a))}{\sum_{k=1}^n \exp(\text{Attention}(e_i, e_k, W_a))}. \tag{4}$$

where the attention module is parameterized by W_a, which is a trainable $d \times d$ weight matrix. The attention score function we adopt is general attention [37]:

$$\text{Attention}(e_i, e_j, W_a) = e_j^\top W_a e_i. \tag{5}$$

The attention module is also parameter shared among agents. We use these attention weights to form an $n \times n$ positive real valued adjacency matrix M with $M_{ij} = \mu_{ij}$, which encodes the implicit coordination graph. Since we apply soft-max for attention weights, we have $\sum_{j=1}^n \mu_{ij} = 1$.

We stack the embeddings to form an $n \times d$ feature matrix E with the ith row being the embedding $e_i^\top$. We denote E as $E^{(0)}$. With the soft adjacency matrix M and the feature matrix $E^{(0)}$, we can apply graph convolution to perform message passing and information integration across all agents. In the fast approximate GCN by Kipf and Welling [31], a graph convolution layer is

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{M}\tilde{D}^{-\frac{1}{2}}H^{(l)}W_c^{(l)}\right), \tag{6}$$

where $H^{(l)}$ is the feature matrix of convolution layer l. In our case, $H^{(0)} = E^{(0)} = [e_1^\top; e_2^\top; \ldots; e_n^\top]$. Diagonal entries of M are already positive from the self-attention weights. Therefore, unlike Kipf and Welling [31], we do not need to add an identity matrix for non-zero self-connections and can set $\tilde{M} = M$. By their definition, $\tilde{D}_{ii} = \sum_{j=1}^{n}\tilde{M}_{ij} = \sum_{j=1}^{n}\mu_{ij} = 1$, i.e. $\tilde{D}$ is simplified to an identity matrix, I_n. The $d \times d$ matrix $W_c^{(l)}$ is a trainable weight matrix associated with layer l, and σ is a non-linear activation.

Replacing $\tilde{M}$ with attention weights M and $\tilde{D}$ with I_n, the graph convolution operation simplifies to

$$H^{(l+1)} = \sigma\left(MH^{(l)}W_c^{(l)}\right). \tag{7}$$

This graph convolution operation is performed m times. We denote the output of mth layer $H^{(m)}$ as $E^{(m)}$, which is a stack of integrated embeddings.

We then use a residual connection [38] between $E^{(0)}$ and $E^{(m)}$ to obtain the final embedding matrix $\tilde{E} = E^{(0)} + E^{(m)}$. The residual connection is designed to assist gradient flow through the attention module and the encoder. The final embedding matrix $\tilde{E}$ consists of a stack of integrated embeddings $\{\tilde{e}_i\}_{i=1}^{n}$. Finally, there are two ways to use the embedding matrix $\tilde{E}$:

(a) DICG-CE: If full communication is allowed between agents, we can use the DICG module in a *centralized-training-centralized-execution* (CTCE) framework to communicate information between the agents. The output from the DICG module, $\tilde{E}$, integrates relevant information across all agents. The corresponding embedding $\tilde{e}_i$ (or its history for recurrent neural networks) can be passed through a separate or parameter-shared policy network to obtain actions for each agent (indicated with blue arrows in Fig. 1). We can then use standard actor-critic methods to train the network end-to-end. As our experiments demonstrate, embeddings obtained from DICG are superior to simply concatenating the raw observations and passing through an MLP network due to the implicit coordination structure reasoning for information integration.

(b) DICG-DE: If full communication is not allowed between agents, we can still use the DICG module to facilitate better coordination. Following the principles of *centralized-training-decentralized-execution* (CTDE), we can use the output of the DICG module, $\tilde{E}$, in a centralized critic. We pass $\tilde{E}$ through another MLP, which we refer to as an aggregator network, to estimate the centralized baseline (indicated with red arrows in Fig. 1). Separate or parameter shared policy networks can be trained for each agent using standard actor-critic methods [2, 19, 39], except using the DICG centralized baseline for advantage computation. During execution the critic is no longer required and the agents can act independently. Again, we find that the embeddings obtained by DICG are superior to simply concatenating the raw observations and passing through an MLP network due to its implicit reasoning about the dynamic coordination structure.

4 Results

We present experiments applying DICG to three environments: predator-prey, StarCraft II Multi-agent Challenge (SMAC) [13], and traffic junction [3]. These environments require coordination to achieve high returns, i.e. agent interactions are not so sparse that totally decentralized approaches with partial observability can achieve high returns. They are also sufficiently complex that fully centralized approaches are intractable. We compare our approach against two standard actor-critic baseline approaches: 1) fully decentralized architecture (referred to as DEC) with only local observation as input to policy; and 2) centralized architecture (referred to as CENT) with a direct concatenation of all observations as input to policy. Due to the dimensionality of the action space, we still need to factorize the policy [2, 3] in the centralized architecture so that we output separate action distributions for each agent. They both use a centralized critic with full observation for baseline estimate, and they use PPO for policy optimization [39]. Benchmarking against them can justify the effectiveness of coordination learning and information integration of DICG. We also compare with results reported by other MARL approaches. All results are averaged over 5 seeds. Shaded area represents 95% CI.

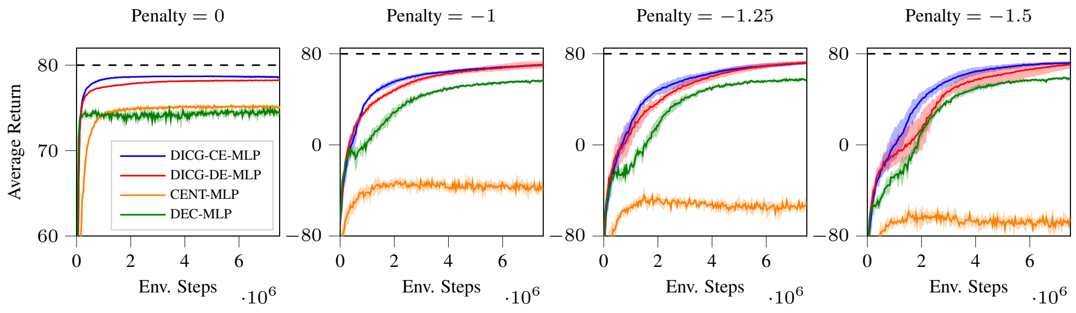

Figure 2: Average return of predator-prey with increasing penalty for single-agent capture attempt.

4.1 Predator-Prey

We use an environment similar to that described by Böhmer *et al.* [10]. The environment consists of a 10×10 grid world with 8 predators and 8 prey. We control the movement of predators to capture prey. The prey move by hard-coded and randomized rules to avoid predators. If a prey is captured, the agents receives a reward of 10. However, the environment penalizes any single-agent attempt to capture prey with a negative reward p; at least two agents are required to be present in the neighboring grid cells of a prey for a successful capture. We set the episode length to 200 steps, and impose a step cost of -0.1. Cooperation is necessary to achieve a high return in this environment. We use a MLP policy for all the architectures. The environment and network details are in Table 4 of the appendix.

Figure 2 shows the average return for test episodes for varying penalties p averaged over 5 runs. Overall, DICG performs the best and solves relative overgeneralization with its implicit coordination. Without any penalty ($p = 0$), fully centralized (CENT-MLP) and fully decentralized (DEC-MLP) architectures have similar performance. However, they require more steps to capture all prey than the DICG approaches. As we increase the penalty, only DICG is able to reliably and quickly converge to optimality. DEC-MLP has a characteristic slowdown in the learning curves before it is able to approach DICG. It finally converges to suboptimal performance with relative overgeneralization due to the lack of coordination across agents. The fully centralized approach CENT-MLP can only achieve positive returns in non-penalized setting. With a negative penalty, CENT-MLP cannot learn to capture prey appropriately due to two reasons: 1) simple concatenation of observations in CENT-MLP leads to a large joint observation space, and 2) concatenation of observations is not an efficient way to integrate information and learn coordination across agents.

Analyzing Implicit Coordination Graph: To understand how the DICG learns to coordinate, we perform attention weight analysis, i.e. we study the strength of soft edges of the implicit coordination. A natural heuristic of what affects the strength of connection between agents is the distance between agents. Fig. 3 shows the relationship between attention weight and distance between agents learned by DICG. Zero distance corresponds to the attention weight of an agent to itself. As the penalty increases, agents tend to increase the attention weight towards agents further away. This phenomenon

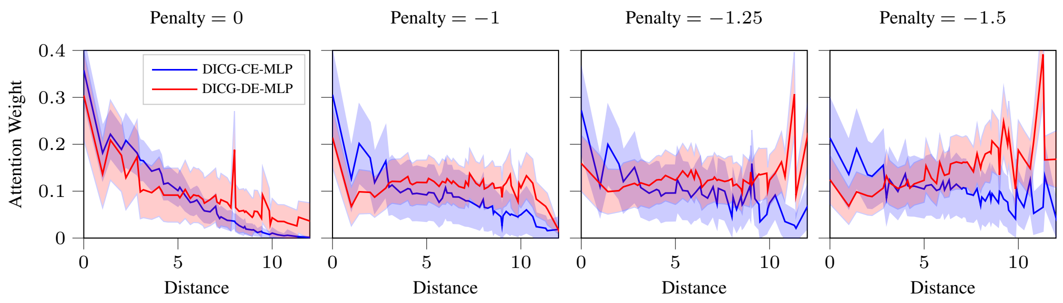

Figure 3: Attention weights, i.e. graph edge strengths of DICG under different distance between two agents (average of 5 seeds). Zero distance indicates an agent's attention towards itself. As the penalty for single-agent capture attempts increases, more attention is paid to farther agents.

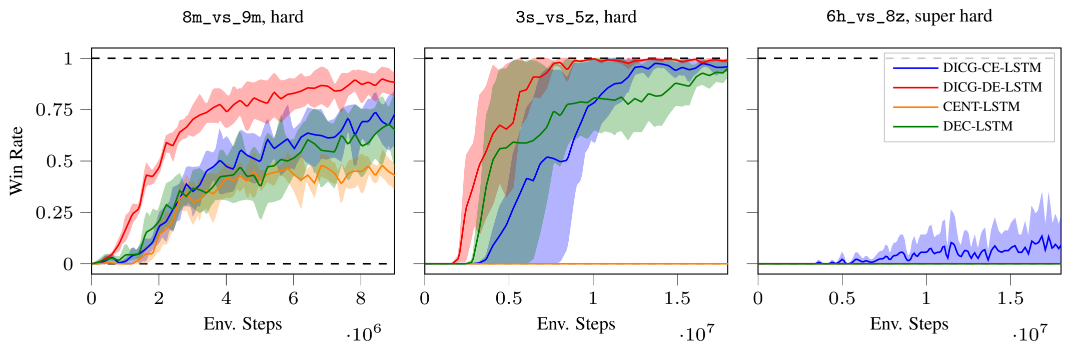

Figure 4: Evaluation win rate during training in SMAC maps.

coincides with the coordination requirements imposed by the increase of penalty that agents should pay more attention to form groups with each other to capture prey as a team than moving alone.

4.2 StarCraft II Multi-agent Challenge (SMAC)

StarCraft II, and the StarCraft II Learning Environment (SC2LE), has provided an environment for some of the most important reinforcement learning work in recent years [40, 41]. However most of this work has resided in the single-agent domain where reinforcement learning is used to train a single, centralized decision-making agent how to play the entire StarCraft II game involving the control of a large number of units within the game. The StarCraft Multi-Agent Challenge (SMAC) extends SC2LE by providing a collection of reinforcement learning benchmarks designed specifically for *multi-agent environments* [13]. Within SMAC, each unit is controlled by its own separate learning agent whose actions must be conditioned on local observations and not the global game state. SMAC scenarios are designed to explore *micromanagement*, e.g. precise movements and coordinated targeting, of relatively small groups of units. In each episode, positive rewards are given for the positive health point difference between the controlled agent team and the computer controlled opponent team, otherwise, the agent team receive zero or negative rewards. Large positive terminal reward is given for winning the episode by eliminating the opponents (zero for being eliminated). Environment and network details can be found in Table 5 and Table 6 respectively in the appendix.

Table 1: SMAC win rate comparison.

Approach	8m_vs_9m	3s_vs_5z	6h_vs_8z
DCG [10]	$55 \pm 10\%$	$85 \pm 3\%$	$\mathbf{10} \pm \mathbf{5}\%$
CENT-LSTM	$42 \pm 6\%$	0	0
DEC-LSTM	$65 \pm 16\%$	$94 \pm 5\%$	0
DICG-CE-LSTM	$72 \pm 11\%$	$96 \pm 3\%$	$9 \pm 9\%$
DICG-DE-LSTM	$\mathbf{87} \pm \mathbf{6}\%$	$\mathbf{99} \pm \mathbf{1}\%$	0

We test DICG on SMAC's asymmetric and "micro-trick" scenarios such as 8m_vs_9m, 3s_vs_5z, and 6h_vs_8z [13]. The opponent AI difficulty is set to "hard", "hard", and "super hard", respectively. We use an LSTM policy for all architectures in SMAC. Results of SMAC are in Fig. 4. In 8m_vs_9m, DICG-DE-LSTM outperforms all the other approaches; DICG-CE-LSTM and DEC-LSTM have similar performance; CENT-LSTM performs the worst. In 3s_vs_5z, DICG-DE-LSTM shows the highest and the most stable win rate, as well as the most sample efficient learning; DICG-CE-LSTM has more stable performance than DEC-LSTM, but CENT-LSTM fails to learn. From game replays, we observe that DICG learns a particular circular movement strategy in 3s_vs_5z. Due to the asymmetric setup, the 3 stalker agents controlled by DICG cannot overcome the opposing 5 zealots with force. The DICG agents learn to split up into two groups, each attracting a number of opponents. Each group moves along the edges of the square map in a circle, and damages opponents using a learned hit-and-run tactic with their high speed. When the opponents are sufficiently weak, the two groups reunite to eliminate the opponents. This highly coordinated tactic demonstrates the effectiveness of DICG. In 6h_vs_8z, a very difficult map, DICG-CE-LSTM is the only approach to win against the opponent AI. A comparison of SMAC win rate under the same difficulty setting with Deep Coordination Graphs (DCG) by Böhmer *et al.* [10] is in Table 1. DICG outperforms DCG *without using the privileged state information* in 8m_vs_9m and 3s_vs_5z, and having comparable but noisier win rate for 6h_vs_8z. In many multi-agent tasks, we do not have access to privileged full state information even during training. DICG demonstrates the advantage of integrated information to prevent relative overgeneralization in such scenarios.

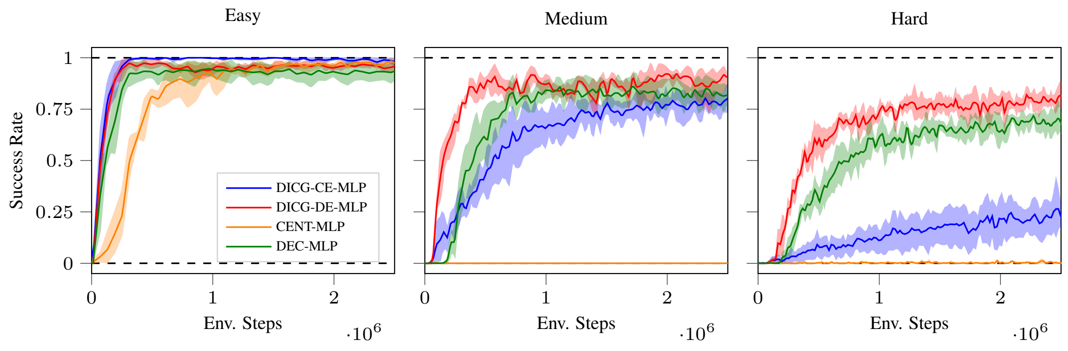

Figure 5: Evaluation success rate during training of different difficulty levels in traffic junction.

4.3 Traffic Junction

The traffic junction environment, originally introduced by Sukhbaatar and Fergus [3], is a multi-agent environment where cars are randomly added to traffic junctions with pre-assigned routes who need to avoid collision with each other and reach their destinations. Each agent only has a limited vision of one grid from itself. The reward function consists of a collision penalty to discourage collision and a step cost to discourage congestion. There are easy, medium, and hard difficulty modes in the traffic junction environment. Their detailed configurations are listed in Table 7 in the appendix. The environment configurations are adopted from Singh *et al.* [42]. We use MLP policies for the traffic junction environment. The network details are in Table 2 of the appendix.

Table 2: Traffic junction success rate comparison.

Approach	Easy	Medium	Hard
CommNet [3]	$93.0 \pm 4.2\%$	$54.3 \pm 14.2\%$	$50.2 \pm 3.5\%$
IC3Net [42]	$93.0 \pm 3.7\%$	$89.3 \pm 2.5\%$	$72.4 \pm 9.6\%$
GA-Comm [35]	$\mathbf{99.7\%}$	$\mathbf{97.6\%}$	$\mathbf{82.3\%}$
CENT-MLP	$97.7 \pm 0.9\%$	0	0
DEC-MLP	$90.2 \pm 6.5\%$	$81.3 \pm 4.8\%$	$69.4 \pm 4.9\%$
DICG-CE-MLP	$\mathbf{98.1 \pm 1.9\%}$	$80.5 \pm 6.8\%$	$22.8 \pm 4.6\%$
DICG-DE-MLP	$95.6 \pm 1.5\%$	$90.8 \pm 2.9\%$	$\mathbf{82.2 \pm 6.0\%}$

The results are in Fig. 5 and a comparison with other baselines using the same environment configurations is in Table 2. DICG-CE-MLP performs better than DICG-DE-MLP in easy mode. CENT-MLP also performs well in easy mode. This is because easy mode has fewer number of agents and small observation space. Centralized execution can outperform decentralized approaches in relatively small domains. However, DICG-CE-MLP has the privilege of more efficient agent information integration over CENT-MLP. In medium and hard mode, where the number of agents and the dimension of observation space increases, centralized approaches fail to perform well. DICG-DE-MLP outperforms decentralized and centralized baselines in medium and hard mode. Even though we *do not use any curriculum* [36] based training, DICG's performance is close to that of GA-Comm [35] which employs curriculum learning. Note that the results of GA-Comm fall within the uncertainty range of our results in easy and hard mode.

5 Conclusions and Future Work

In this work, we present the DICG architecture that uses self-attention to implicitly build a coordination graph and then perform message passing with graph convolution layers to compute appropriate baseline values (DE) or actions (CE) for the agents while keeping the computational graph differentiable. We demonstrate that DICG solves the relative overgeneralization pathology in predator-prey tasks, as well as various MARL baselines including the challenging StarCraft II micromanagement tasks and traffic junction tasks. DICG is shown to be an effective architecture for implicitly and dynamically learning multi-agent coordination that achieves an appropriate tradeoff between fully centralized and fully decentralized approaches. For future work, we aim to improve the sample efficiency of DICG. To achieve this, we may incorporate the DICG architecture into off-policy learning algorithms such as deep Q-learning [43] and soft actor-critic [20, 44] for multi-agent problems.

Broader Impact

Coordination is important in a variety of multi-agent tasks such as autonomous driving, logistics, package delivery, etc. The approach suggested in this paper could also be useful for deploying resources for wildlife protection against illegal poaching and other nature conservation efforts [45–47]. As with other multi-agent research, this approach can also be used for planning in battlefield environments [48].

References

[1] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *International Conference on Machine Learning (ICML)*, 1993, pp. 330–337.

[2] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Springer, 2017, pp. 66–83.

[3] S. Sukhbaatar and R. Fergus, "Learning multiagent communication with backpropagation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016, pp. 2244–2252.

[4] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[5] L. Panait, S. Luke, and R. P. Wiegand, "Biasing coevolutionary search for optimal multiagent behaviors," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 629–645, 2006.

[6] C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored MDPs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2002, pp. 1523–1530.

[7] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 2085–2087.

[8] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2018, pp. 4295–4304.

[9] J. R. Kok and N. Vlassis, "Sparse cooperative Q-learning," in *International Conference on Machine Learning (ICML)*, ACM, 2004, p. 61.

[10] W. Böhmer, V. Kurin, and S. Whiteson, "Deep coordination graphs," *arXiv preprint arXiv:1910.00091*, 2019.

[11] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2020.

[12] J. R. Kok, E. J. Hoen, B. Bakker, and N. Vlassis, "Utile coordination: Learning interdependencies among cooperative agents," in *IEEE Symposium on Computational Intelligence and Games*, 2005, pp. 29–36.

[13] M. Samvelyan, T. Rashid, C. Schroeder de Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, "The Starcraft multi-agent challenge," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 2186–2188.

[14] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[16] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[19] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[20] E. Wei, D. Wicke, D. Freelan, and S. Luke, "Multiagent soft Q-learning," in *AAAI Spring Symposium*, 2018.

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

[22] A. OroojlooyJadid and D. Hajinezhad, “A review of cooperative multi-agent deep reinforcement learning,” *arXiv preprint arXiv:1908.03963*, 2019.

[23] N. Vlassis, R. Elhorst, and J. R. Kok, “Anytime algorithms for multiagent decision making using coordination graphs,” in *IEEE International Conference on Systems, Man and Cybernetics*, 2004, pp. 953–957.

[24] J. Cheng, L. Dong, and M. Lapata, “Long short-term memory-networks for machine reading,” *arXiv preprint arXiv:1601.06733*, 2016.

[25] X. Liu, H. Li, J. Shao, D. Chen, and X. Wang, “Show, tell and discriminate: Image captioning by self-retrieval with partially labeled data,” in *European Conference on Computer Vision (ECCV)*, 2018, pp. 338–354.

[26] J. Yu, J. Li, Z. Yu, and Q. Huang, “Multimodal transformer with multi-view visual representation for image captioning,” *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[27] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le, “QANet: Combining local convolution with global self-attention for reading comprehension,” *arXiv preprint arXiv:1804.09541*, 2018.

[28] S. Iqbal and F. Sha, “Actor-attention-critic for multi-agent reinforcement learning,” *arXiv preprint arXiv:1810.02912*, 2018.

[29] J. Jiang and Z. Lu, “Learning attentional communication for multi-agent cooperation,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 7254–7264.

[30] M. A. Wright and R. Horowitz, “Attentional policies for cross-context multi-agent reinforcement learning,” *arXiv preprint arXiv:1905.13428*, 2019.

[31] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.

[32] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” in *International Conference on Learning Representations (ICLR)*, 2018.

[33] A. Malysheva, T. T. Sung, C.-B. Sohn, D. Kudenko, and A. Shpilman, “Deep multi-agent reinforcement learning with relevance graphs,” *arXiv preprint arXiv:1811.12557*, 2018.

[34] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, *et al.*, “Relational deep reinforcement learning,” *arXiv preprint arXiv:1806.01830*, 2018.

[35] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, “Multi-agent game abstraction via graph attention neural network,” *arXiv preprint arXiv:1911.10715*, 2019.

[36] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in *International Conference on Machine Learning (ICML)*, 2009, pp. 41–48.

[37] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *arXiv preprint arXiv:1508.04025*, 2015.

[38] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[39] R. E. Allen, J. W. Bear, J. K. Gupta, and M. J. Kochenderfer, “Health-informed policy gradients for multi-agent reinforcement learning,” *arXiv preprint arXiv:1908.01022*, 2019.

[40] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, *et al.*, “Starcraft II: A new challenge for reinforcement learning,” *arXiv preprint arXiv:1708.04782*, 2017.

[41] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.*, “Grandmaster level in starcraft II using multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[42] A. Singh, T. Jain, and S. Sukhbaatar, “Learning when to communicate at scale in multiagent cooperative and competitive tasks,” *arXiv preprint arXiv:1812.09755*, 2018.

[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[44] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International Conference on Machine Learning (ICML)*, 2018, pp. 1861–1870.

[45] N. Kamra, U. Gupta, F. Fang, Y. Liu, and M. Tambe, “Policy learning for continuous space security games using neural networks,” in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[46] Y. Wang, Z. R. Shi, L. Yu, Y. Wu, R. Singh, L. Joppa, and F. Fang, “Deep reinforcement learning for green security games with real-time information,” in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 33, 2019, pp. 1401–1408.

[47] F. Fang, “Integrate learning with game theory for societal challenges,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, AAAI Press, 2019, pp. 6393–6397.

[48] A. Feickert, J. K. Elsea, L. Kapp, and L. A. Harris, *US ground forces robotics and autonomous systems (RAS) and artificial intelligence (AI): Considerations for Congress*. Congressional Research Service, 2018.

[49] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[50] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems (NeurIPS)*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.