Cyclicity-Regularized Coordination Graphs

Oliver Järnefelt[1], **Mahdi Kallel**[1,*], **Carlo D'Eramo**[1,2,3]
[1]Center for Artificial Intelligence and Data Science, University of Würzburg, Germany
[2]Department of Computer Science, TU Darmstadt, Germany
[3]Hessian Center for Artificial Intelligence (Hessian.ai), Germany

Abstract

In parallel with the rise of the successful value function factorization approach, numerous recent studies on Cooperative Multi-Agent Reinforcement Learning (MARL) have explored the application of Coordination Graphs (CG) to model the communication requirements among the agent population. These coordination problems often exhibit structural sparsity, which facilitates accurate joint value function learning with CGs. Value-based methods necessitate the computation of argmaxes over the exponentially large joint action space, leading to the adoption of the max-sum method from the distributed constraint optimization (DCOP) literature. However, it has been empirically observed that the performance of max-sum deteriorates with an increase in the number of agents, attributed to the increased cyclicity of the graph. While previous works have tackled this issue by sparsifying the graph based on a metric of edge importance, thereby demonstrating improved performance, we argue that neglecting topological considerations during the sparsification procedure can adversely affect action selection. Consequently, we advocate for the explicit consideration of graph cyclicity alongside edge importances. We demonstrate that this approach results in superior performance across various challenging coordination problems.

1 Introduction

The ability for autonomous agents to collaborate is crucial, spanning applications from multi-robot systems [7] to sensor networks [6, 16, 12]. The quest for learning effective control policies with multi-agent reinforcement learning (MARL) [18] mirrors the strategies employed in single-agent environments, but presents unique challenges. While learning individual action-value functions [26] is scalable, it suffers from the issue of non-stationarity caused by the inability to predict other agents' behaviour. On the other hand, joint action-value learning [4] mitigates non-stationarity but requires often unavailable global information, and becomes intractable with the number of agents due to the exponentially large joint action space. Recently, there has been a strong emphasis on value function factorization (VFF) methods that construct the joint action-value function as a mixing of individual agent utilities [25, 20, 21, 24, 27]. However, lacking mechanisms to explicitly model coordination, VFF methods are shown to suffer from the *relative overgeneralization pathology* [1].

Tackling these problems, the formalism of Coordination Graphs (CG) [8] has experienced a recent renaissance. CGs constitute an interpretable graphical model representing the state-dependent coordination structure and induce a factorization of the joint action-value function as a sum of single-agent utility functions and payoff functions for agent pairs. Whilst incorporating pairwise payoffs into the learning process mitigates the non-stationarity issue, the use of max-sum algorithm [19] circumvents the scaling problems ailing joint learning by providing an approximation for the greedy joint action selection. Deep Coordination Graphs (DCG) [1] integrate CGs into the MARL framework and show the pairwise payoffs form crucial mechanism to overcome relative overgeneralization.

[*]Correspondence to `mahdi.kallel@uni-wuerzburg.de`.

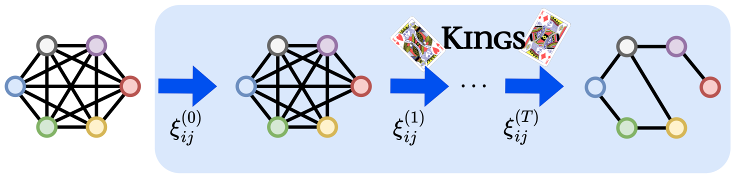

Figure 1: KINGS constructs the sparse coordination graph by iteratively considering the best edge to prune based on the mutual influence scores and the current graph topology.

While DCG assumes a fixed structure for the coordination graph, Wang et al. [29] argue that in a wide variety of problems the communication requirements are dynamic and sparse in terms of considering only a subset of pairwise agent relations. They introduce a CG sparsification method that works by ranking the payoff functions by a measure quantifying the influence between the agents belonging to the edge. In this work, we reconsider said metric by demonstrating that especially with high sparsity levels, omitting topological information related to the graph, has a negative impact on the max-sum quality arising from graph disconnectivity and excessive cyclicity. The latter has been shown to be especially detrimental to the max-sum algorithm [15, 3]. Towards mitigating these issues, we propose a novel cyclicity aware sparsification metric for sparse CGs and demonstrate its positive effect on sparse CG-based MARL with an evaluation on the Multi-Agent Coordination Benchmark (MACO) [29] and the StarCraft Multi-Agent Challenge (SMAC) [23].

2 Background

The cooperative problems we consider in this work fall under the framework of DEC-POMDPs [17]. A DEC-POMDP is described via a tuple $\mathcal{M} := \langle I, N, \mathcal{S}, \mathcal{A}, T, R, \Omega, O, \gamma \rangle$, where I is the set of N agents, $\mathcal{S}$ the state space, $\mathcal{A}$ the action space shared by all of the agents, T the joint transition kernel, R the joint reward function, Ω the observation space, O the observation function and γ the discount factor. For a given DEC-POMDP, our objective is to find a mutually independent set of policies $\boldsymbol{\pi}^*$ satisfying: $\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{a} \sim \boldsymbol{\pi}} Q^*_{\text{jt}}(s, \boldsymbol{a})$, where $Q^*_{\text{jt}}(s, \boldsymbol{a})$ denotes the total expected future discounted returns or the *optimal joint utility function*.

2.1 Coordination Graphs and Max-Sum

The problem of learning the joint utility function scales very poorly in the number of agents belonging to the task due to the exponential size of the joint action space. Luckily, most meaningful multi-agent problems can be effectively represented by considering only pairwise interactions. To this end, Guestrin et al. [8] introduce the concept of *Coordination Graphs*. A Coordination Graph (CG) $\mathcal{G} := \langle \mathcal{V}, \mathcal{E} \rangle$ specifies the pairwise communication requirements between the agents $\mathcal{V}$ via its edges $\mathcal{E}$. It induces a factorization of the total utility function into

$$Q_{\text{jt}}(\boldsymbol{\tau}, \boldsymbol{a}) = \frac{1}{N} \sum_{i=1}^{N} f_i(a_i|\tau_i) + \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} f_{ij}(a_i, a_j|\tau_i, \tau_j).$$

Representing the total utility with CGs enables the use of the max-sum algorithm [6] for the greedy action selection that becomes intractable to perform via exhaustive enumeration in multi-agent settings. Max-sum is known to converge only for acyclic graphs but has been applied comprehensively in also problems exhibiting varying levels of cyclicity.

Böhmer et al. [1] adapt the concept of CGs to Multi-Agent Deep Reinforcement Learning (MARL), by parameterizing the set of utility and payoff functions f_{ij} with neural networks. Their approach, known as Deep Coordination Graphs (DCG), trains each of the utility and payoff networks with temporal difference learning in the style of DQN [14]. Whereas DCG proposes to train the joint utility by considering the full set of pairwise interactions in each state, Kok and Vlassis [11] argue that many practical coordination problems exhibit a certain level of sparsity, meaning that one can instead focus on a sparse subset of pairwise factors, thus improving learning speed. This concept is first explored in deep MARL by Wang et al. [29] who propose a state-dependent mechanism to selectively filter out a relevant subset of edges for more efficient message passing in max-sum. Their approach, CASEC, suggests identifying the best edges in terms of their influence on the task at hand, which can be defined as: $\zeta_{i\to j} = \max_{a_i} \text{Var}_{a_j} f_{ij}(a_i, a_j|\tau_i, \tau_j)$. Then, for an undirected edge $e = (i, j)$, we write the undirected *mutual influence* of the agents as

$$\zeta_{ij} = \max\{\zeta_{i\to j}, \zeta_{j\to i}\}. \tag{1}$$

CASEC sparsification requires defining a *sparsity coefficient* λ_{sp}, which takes values in the interval $[0, 1]$ and determines the proportion of edges in the original fully connected coordination graph to be pruned. CASEC then ranks the edges according to their ζ-value and retains $(1 - \lambda_{\text{sp}})(N^2 - N)/2$ top scoring edges to be used by max-sum. While this simple modification demonstrates promising results in tasks exhibiting sparsity, we argue that ignoring the topological changes caused by the sparsification have detrimental effects on the max-sum accuracy that in turn hinders learning.

3 Related Works

Max-sum has been studied in the distributed constraint optimization (DCOP) literature, and a reasonable amount of attention has been given to the algorithm's lack of convergence guarantees with cyclic graphs and its practical implications [5]. Various methods have been proposed which either act by modifying either the graph max-sum takes in [15, 22] or modifying max-sum itself [3]. Such considerations have been considered also in the context of MARL. In addition to the work of sparsifying CGs with mutual influence metrics [29], Yang et al. [31] have proposed limiting the joint model's representational capacity to only directed acyclic graphs. In this work, we instead want to retain the possibility for cycles to exist and ask how to mitigate the negative effects of cyclicity.

In parallel to the advances made with CGs, there has been a recent surge of research activity around *value function factorization* (VFF) [28]. VFF approaches train a centralized value function, a parametric *mixing* of the individual agent utilities, to produce an estimate for the joint utility. When the gradient signal originating from the environmental reward gets backpropagated, said mixing acts as an implicit mechanism to distribute the credit amongst the population of agents. Many previous works have focused on finding the most flexible way to parametrize the joint value function while ensuring decomposability. The choice of the mixing function determines the representational capacity of the joint utility model and has large implications on the performance. Value-Decomposition Networks (VDN) [25] consider only additive mixing functions, whereas QMIX [20] extends to the whole family of monotonic functions. More recently, a plethora of other VFF methods have been proposed and methods that cover the entire Independent-Global-Max (IGM) space have been introduced [24, 27].

4 Cyclicity–Regularized Coordination Graphs

In the upcoming section, we present our contribution to the learning of cyclicity-regularized sparse coordination graphs. Section 4.1 introduces and justifies our proposed method, and Section 4.2 offers an illustrative demonstration to provide additional motivation.

4.1 Kirchoff Index Guided Sparsification

While the ζ-metric, defined in Eq. (1) is appealing in that it ties the edge importance to the variation in the payoff functions, we argue that such sparsification criterion can be problematic for two reasons. First, it has been observed that the performance of max-sum deteriorates when increasing the number of agents due to increased overall cyclicity of the graph [15, 22, 3], which is overlooked by ζ-sparsification. Secondly, when the sparsity level is set high, many nodes can become disconnected from the rest of the coordination graph impairing communication. This can be especially catastrophic in situations where all agents need to have knowledge of others' intentions to make a decision for themselves. Motivated by these issues, we propose that one should instead search for sparse graphs that, in addition to valuing edges with high mutual influence, 1) retain connectedness ensuring proper flow of information between all agents and 2) are minimally cyclic. Towards these objectives, we propose to iteratively sparsify the graph edges according to a modified mutual influence metric

$$\xi_{ij} = \frac{\zeta_{ij}}{A_{ij} - \Omega_{ij}}, \tag{2}$$

where A_{ij} is the adjacency matrix of the CG and Ω_{ij} is called the *effective resistance* matrix [10]. The effective resistance is a quantity that describes the commute time of a random walk between two nodes i and j in a resistor network whose topology is given by A and each edge is replaced with a unit resistor. Let us denote the graph Laplacian as L and a full ones matrix of shape $N \times N$ as Φ. With these definitions we can compute the entries of Ω as

$$\Omega_{ij} = \Gamma_{ii} - \Gamma_{jj} - 2\Gamma_{ij} \quad \text{where} \quad \Gamma = \left(L + \frac{1}{N}\Phi\right)^{-1}. \tag{3}$$

When the effective resistance Ω_{ij} is compared to the *nominal* resistance given by A_{ij} as in the denominator of Eq. (2), one obtains a quantity called the *resistance deficit* which has been used in graph theory to assess an edge's contribution to the total cyclicity of a graph [9, 32]. Thus, resistance deficit penalization weights down the edge scores proportional to how much they increase the total cyclicity of the graph. As Γ changes every time an edge is pruned, computing the final sparse graph must be done by computing current ξ_{ij}, dropping the edge with the lowest ξ-value, repeating until target sparsity is reached. Such iterative procedure aims to maximize the Kirchhoff Index [13] of the final graph. The following proposition, establishes a theoretical justification for the proposed approach.

Proposition 1. *CG sparsification with the ξ-metric is guaranteed to keep the graph connected for $\lambda_{sp} \in [0, 1 - 2/N]$. When $\lambda_{sp} \in [1 - 2/N, 1]$, the final graph be a forest enabling max-sum to converge to a fixed point in finite number of iterations.*

Proof. Assume input graph $\mathcal{G} := \langle \mathcal{V}, \mathcal{E} \rangle$. When the final graph sparsity is $\lambda_{\text{sp}} \in \left[0, 1 - \frac{2}{N}\right]$, the graph is guaranteed to be connected. To see this, we analyze the graph topology at iteration t of the iterative sparsification procedure. Let us divide the nodes in two sets: the ones with only 1 edge connected to them and the ones that have more than 1 edge connected and mark the sets $\mathcal{V}_-$ and $\mathcal{V}_+$, respectively. For edge $(i, j),\ i \in \mathcal{V}_-, j \in \mathcal{V}_- \cup \mathcal{V}_+$, we have that $A_{ij} - \Omega_{ij} = 0 \implies \xi_{ij} \to \infty$. In words, when an edge describes the only possible path between two nodes, the corresponding *resistance deficit* for that edge is 0, leading to an unbounded ξ-value. As a result, only edges connected exclusively to $\mathcal{V}_+$, will have bounded ξ-values. As also the lowest scoring edge is sparsified, $\mathcal{V}_-$ will still remain connected to the rest of the graph.

A direct consequence of the connectedness is that when $\lambda_{\text{sp}} = 1 - \frac{2}{N}$ the final sparse graph will be a spanning tree for $\mathcal{G}$. Even further increasing sparsity thus leads into a spanning forest for $\mathcal{G}$. Using the results from Pearl [19], we can also note that max-sum on graphs with $\lambda_{\text{sp}} \geq 1 - 2/N$ is guaranteed to converge.

$\square$

Algorithm 1 KINGS

```
Input: $N$, $f_{ij}$, $\lambda_{\text{sp}}$, budget $I$, adjacency $A_{ij}$
  $\zeta_{ij} \leftarrow$ mutual_influence($f_{ij}$)   # Eq. (1)
  $M_{ij} \leftarrow 1_{N \times N}$
  $C \leftarrow \lambda_{sp} \frac{N^2 - N}{2 \cdot I}$
  if $C > 1$ then
      $M \leftarrow$ NOT(max_spanning_tree($\zeta_{ij}$))
  for $i \in [0, \dots, I]$ do
      $\Omega_{ij} \leftarrow$ eff_resistance($A_{ij}$)   # Eq. (3)
      $\xi_{ij} \leftarrow \zeta_{ij} / ((A_{ij} - \Omega_{ij}) \odot M)$
      # Prune $C$ worst scoring edges
      $e^- \leftarrow$ topk($-\xi_{ij}, k = C$)
      $A_{ij} \leftarrow$ remove($A_{ij}, e^-$)
  return $A_{ij}$
```

Importantly, this result shows that ξ-sparsification leads to desirable graphs – ones that are connected when sparsity level allows, or ones that are optimal from the perspective of max-sum. While the convergence guarantees can be given when $\lambda_{\text{sp}} > 1 - 2/N$, we also expect that simply *reducing* the amount of cyclicity in the graph can be helpful for max-sum based on the results of Cerquides et al. [3]. We verify this supposition in the subsequent subsection by examining the max-sum accuracy as a function of the sparsity coefficient. Finally, by incorporating ζ in the construction, ξ-graphs also aim to select graphs that are maximally influential.

While modulating the original ζ-scores with structural information is beneficial, we acknowledge that this procedure can become costly when graph sizes grow. This is because we are required to recompute Γ at each iteration, which involves inverting the $N \times N$ graph Laplacian. In order to scale the cyclicity minimizing sparsification to deep MARL, we instead sparsify the original coordination graph in *chunks* of edges: instead of pruning the graph one edge at a time, we fix the sparsification iteration budget I, and at each iteration prune as many edges are needed to reach the target sparsity within the required number of iterations. This way we are trading off the accuracy of assessing the graph cyclicity for computation speed. To ensure final graph connectedness, we require that the final graph contains the edges that belong to the maximum spanning tree (MaxST) of edge weights ζ_{ij}, which we expect to be an important path in terms of utility estimate communication carried out by max-sum. For further discussion on the sparsification budget, we refer the reader to Section 5.3. We call this final version of the iterative sparsification procedure Kirchhoff Index Guided Sparsification (KINGS). Pseudocode 1 summarizes the algorithm.

4.2 Motivating Example

To illustrate the effect of the modified sparsification metric, we randomly generate 1000 fully connected coordination graphs. The exact generation process is described in detail in Appendix A. Each graph features 11 agents, all of which can take 3 distinct actions. We then heavily sparsify all of the graphs by sliding λ_{sp} over the interval $[0.7, 1]$ using both ζ- and ξ-metrics. We will refer to the graphs sparsified with ξ and ζ as ξ-graphs and ζ-graphs, respectively. To measure the impact of sparsification on max-sum, we exhaustively search for the maximum Q-value and the corresponding optimal action for each of the sparsified graphs. Then, we compare the result to the action selected by running max-sum on the corresponding graph for 100 iterations to compute the max-sum accuracy. Figure 2 shows that ξ-graphs enjoy a markable edge in max-sum accuracy compared to ζ-graphs, hinting at the benefit of integrating topological information into the pruning procedure. As expected from previous discussion, the accuracy for ξ-graphs reaches 1 at $\lambda_{\text{sp}} = 9/11 \approx 0.82$, which is also when the sparsified graph is a spanning tree for the 11 nodes, in contrast with ζ-graphs that still sustain over 5% error for the same level of sparsity.

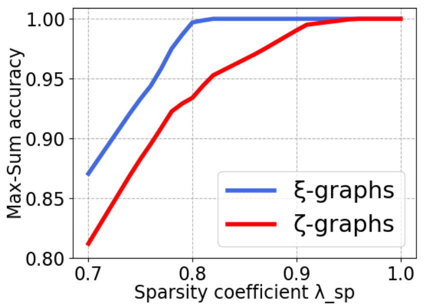

Figure 2: Max-sum greedy action selection accuracy on random CGs sparsified with either ζ- or ξ-metrics

5 Experiments

To understand the effectiveness of the our method, we evaluate KINGS in two cooperative multi-agent task suites, Multi-Agent Coordination Benchmark (MACO) [29] and Starcraft Multi-Agent Challenge (SMAC) [23]. The key questions we aim to answer are: **(I)** How does KINGS fare against the state-of-the-art cooperative MARL methods in difficult coordination tasks? **(II)** Fixing communication bandwidth, does cyclicity-awareness improve sparse CG learning? **(III)** What qualitative insights can we gain regarding the scenarios in which pruning with topological information proves advantageous? **(IV)** What is the impact of the sparsification budget assumed by KINGS? For all of the results in this paper, we plot the mean performance along with the standard error bounds computed over 5 random seeds.

The overall hyperparameters are as in the previous work [25, 20, 1, 29]. All tasks employ a discount factor $\gamma = 0.99$. Each network is trained using an RMSProp optimizer with a learning rate of 5×10^{-3}. A first-in-first-out (FIFO) replay buffer stores the experiences of at most 5000 episodes, and a batch of 32 episodes are sampled from the buffer during the training phase. The target network undergoes periodic updates every 200 episodes. We implement ϵ-greedy exploration, with ϵ linearly annealing from 1.0 to 0.05 over $50K$ time-steps. To ensure a fair comparison, our method and all the baselines presented in this paper are implemented using the open-sourced codebase PyMARL [23]. All the CG based methods perform 5 iterations for max-sum.

5.1 Performance Evaluation

5.1.1 MACO

We begin the performance evaluation with a focus on the Multi-Agent Coordination Benchmark (MACO) [29], which consists of temporally extended versions of 6 classical coordination games compiled in Castellini et al. [2]. The reward structure in each environment is either factored or non-factored, depending on whether an explicit decomposition of global rewards is present. Crucially,

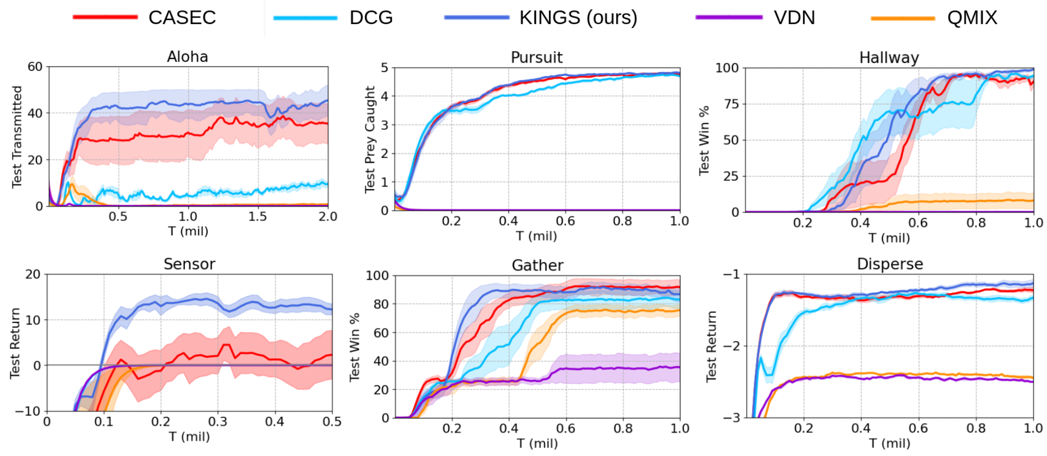

Figure 3: Performance evaluation of KINGS in the MACO benchmark against the baseline methods.

MACO tasks test the algorithms' ability to overcome the relative overgeneralization pathology [1] and sustain cooperation that extends over multiple timesteps. As relevant baselines, we demonstrate the results of VDN [25] and QMIX [20] for value function factorization methods and DCG [1] and CASEC [29] for coordination graph-based approaches.

Table 1 summarizes the sparsity coefficients as percentages for CASEC and KINGS in each environment of MACO. For CASEC, we used the λ_{sp} values provided in the original paper, whereas for KINGS we find the best sparsity coefficient that is greater or equal to the one of CASEC. However, as we want to retain connectedness, we stop at $1 - 2/15 \approx 0.867$ in `Sensor`. The sparsification budgets are chosen from the range $[0, 3]$ ensuring that the computation times stay within reasonable limits. The exact values used are provided in Table 2. The results for the MACO environments are shown in Figure 3. Both QMIX and VDN perform very badly due to their inability to overcome the relative overgeneralization pathology. In contrast, all the CG-based approaches provide much better results in these environments. Second point to note is that in most of the environments enforcing sparse inputs to max-sum seems beneficial as demonstrated by both KINGS and CASEC in relation to DCG. Convincingly, there is a noticeable difference between the performances of KINGS and CASEC, supporting our claim on the importance of topological information in sparsification. While the results of CASEC and KINGS are rather similar on the `Gather` environment, it is crucial to consider that CASEC is able to achieve such solution only when it is allowed to retain 70% of the original graph edges. In contrast we achieve similar final performance with only 40% of the original edges, hinting that KINGS enables the population to learn more effective patterns of communication.

Aloha	**Pursuit**	**Hallway**	**Sensor**	**Gather**	**Disperse**
80%	70%	50%	87%|90%	60%|30%	60%

Table 1: Sparsity coefficients used on MACO. Blue for KINGS, Red for CASEC, Black for both.

5.1.2 SMAC

To get a more thorough understanding of the utility of integrating topological information to the sparsification on learning as well as the scalability to very complex multi-agent problems, we test KINGS against the same baselines on SMAC benchmark [23] which is based on the real time strategy game StarCraft II. As the results are not directly comparable across different versions of StarCraft, we use the version 2.4.6.2.69232 specified in the original SMAC paper. Specifically, we benchmark the different methods on 2 SMAC maps: `1c3s5z` and `10m_vs_11m`. We employ $\lambda_{\text{sp}} = 0.65$ in `10m_vs_11m` and $\lambda_{\text{sp}} = 0.75$ in `1c3s5z` for both KINGS and CASEC. These were the highest values of sparsity for which either of the sparse CG methods worked reasonably well in our tests, and they were found by performing a rough sweep over the values $\{0.3, 0.4, 0.5, 0.65, 0.75\}$ for both maps. The sparsification budget for KINGS is set to 2 in both of the SMAC environments.

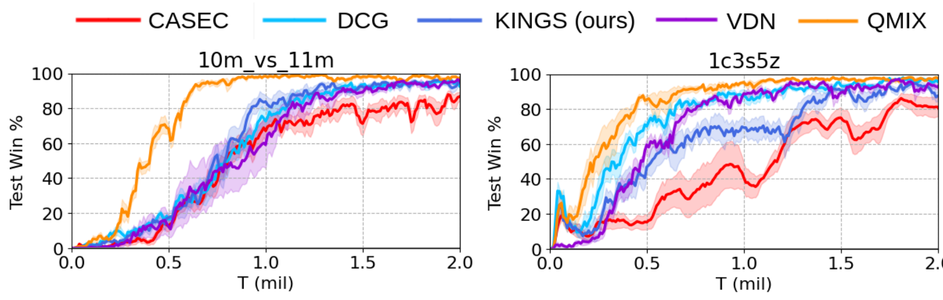

Figure 4: Performance on SMAC. The sparsity coefficients for KINGS and CASEC are 0.65 and 0.75 for `10m_vs_11m` and `1c3s5z`, respectively.

As observed by Böhmer et al. [1], Wang et al. [29], Yang et al. [31], learning the payoff functions f_{ij} is difficult due to the quadratic number of output heads in the output layer. Performing Q-learning style training with such networks can cause many of the output connections to remain unchanged for long periods which causes inaccuracies in the Q_{jt}-estimation. This problem is exacerbated by the fact that ζ-metric used both by KINGS and CASEC relies on the accuracy of the learned payoff functions. To circumvent this problem, we apply action representation learning [29] when training KINGS, CASEC and DCG. We also present impact of this architectural modification for MACO environments in Appendix C.

Figure 4 presents the results on the tested environments. In contrast to MACO environments, fully decomposed methods VDN and QMIX demonstrate strong performance on each of the tasks. Despite the apparent lack of performance enhancement due to sparsity enforcement, a notable distinction emerges in the sparsity tolerance of KINGS when compared to CASEC. KINGS maintains higher performance under a fixed sparsity level, implying a more effective communication capability in these environments.

5.2 Tolerance to Sparsity

As demonstrated in earlier sections, KINGS consistently outperforms CASEC when the sparsity is fixed. Here, we reinforce this observation with additional results, selecting `Hallway` and `Gather` as illustrative examples of factored and non-factored MACO environments. For both, we study the final performances of KINGS and CASEC in the range of sparsity coefficients $[\lambda_{\text{sp}}^{\text{lo}}, \lambda_{\text{sp}}^{\text{hi}}]$, where $\lambda_{\text{sp}}^{\text{lo}}$ are optimal values for CASEC and $\lambda_{\text{sp}}^{\text{hi}}$ are the values after which CG can't be kept connected. We discretize these ranges into 4 values and plot the results for each one. Figures 5 demonstrate that KINGS is able to better withstand sparsity, especially in `Gather` where performance gradually increases with λ_{sp}, giving support to the idea that reducing cyclicity can help learning in max-sum based approaches. While KINGS remains unaffected by sparsity up to $\lambda_{\text{sp}} = 0.7$ in `Hallway`, Figure 6 reveals that both methods suffer from increased sparsity in their learning speeds to a similar extent.

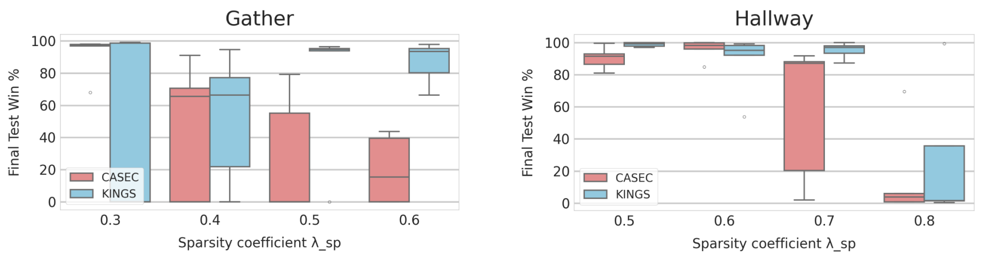

Figure 5: The effect of λ_{sp} on KINGS and CASEC in `Gather` and `Hallway` environments. In most cases, KINGS achieves a higher performance for a given level of sparsity.

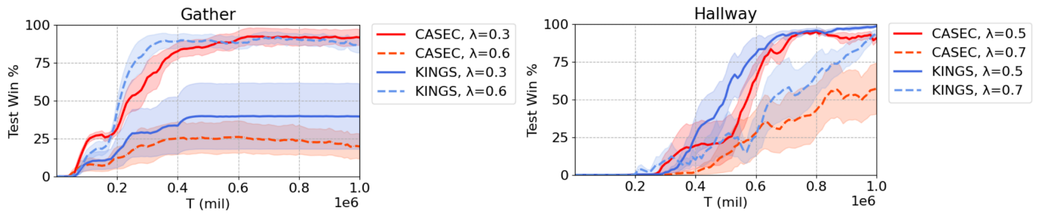

Figure 6: Performance for two sparsity levels for KINGS and CASEC in `Hallway` and `Gather`. The advantage of KINGS is clear in `Gather`, where the CG connectedness is integral to solving the task.

Aloha	**Pursuit**	**Hallway**	**Sensor**	**Gather**	**Disperse**
0	2	3	0	6	2

Table 2: Sparsification budgets used in the MACO evaluation.

5.3 Effect of Sparsification Budget

Like stated in Section 4.1, the practical implementation of KINGS needs a fixed sparsification budget. Algorithmically, increasing the budget means that one has more iterations to reach the target sparsity λ_{sp}, which enables pruning the graph in a more fine-grained manner. Contrarily, decreasing the budget means that in order to reach the target sparsity, one needs to sparsify more edges at once. As mentioned, Table 2 shows the choices of the sparsification budgets in different MACO environments. In `Aloha` and `Sensor`, we are sparsifying the fully connected graph down to a spanning tree, which can be seen from their corresponding λ_{sp}, and thus we directly take the maximum spanning tree – this is equivalent to running KINGS with sparsification budget 0. In `Gather` we afford to apply the procedure edge by edge due to the small size of the fully connected graph, which has $(5^2 - 5)/2 = 10$ edges, thus 6 sparsification iterations. For `Pursuit` and `Disperse` we tested both budgets 2 and 3 observing little difference between them two choices.

The only environment, where we observed the chosen budget to have a more noticeable impact was `Hallway`, with $\lambda_{\text{sp}} = 0.5$. The chosen sparsity coefficient translates to pruning 33 edges from the original fully connected CG. In this setting, we tested KINGS with budgets $1, 2$ and 3. This corresponds to sparsifying the original graph in chunks of sizes 33, $16/17$ or 11, respectively. The results are shown in Figure 7. While the distribution of final performances is about the same for all of the tested budgets, we see that the learning speed is slightly affected as we make the pruning coarser. Conversely, tightening the budget from 1 to 2 translates to about 20% speedup in the *total* run time while retaining. Additionally, Table 3 presents the comparisons for the speeds between CG-based methods over 1000 action selections. This observation provides support for the supposed trade-off between the sparsification accuracy and computational requirements for this environment. An interesting avenue for future work would be to study formally the changes different sparsification budgets have on the final graph cyclicity.

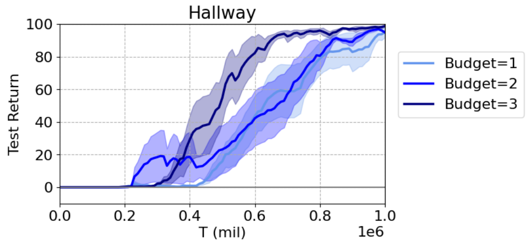

Figure 7: The effect of the sparsification budget on Hallway.

DCG	**CASEC**	**KINGS**, budget=1	**KINGS**, budget=2	**KINGS**, budget=3
2.2	2.5	2.9	3.1	3.4

Table 3: Time taken in seconds for 1000 action selections with DCG, CASEC and KINGS with different sparsification budgets in `Hallway` and $\lambda_{\text{sp}} = 0.5$.

5.4 Qualitative Analysis on Gather

To conclude the experiments, we emphasize the significance of topological information through a specific case in `Gather`, which extends classical cooperative Climb Game [30] by introducing temporal complexity and stochasticity, requiring agents to learn cooperative *policies* instead of atomic actions. The map for the environment is a 3×5 grid, where agents spawn randomly. In each episode, a target goal for the population is randomly selected among three goal states: g_1, g_2 and g_3. These three states are all placed on the 2nd row, at 1st, 3rd and 5th columns, respectively. Agents near the target goal are the only ones aware of its optimality and thus need to communicate this knowledge to the others. The episode ends either after 8 steps or when all agents have gathered to a single goal state. To earn a reward, all agents must simultaneously be located at some goal state. Achieving the target goal results in a high reward of 10, gathering at other goals yields 5, and a minimum reward of -5 is assigned if only some agents gather at the optimal goal, increasing the difficulty. Thus, solving the environment implies maintaining CG connectedness as all agents must reach their designated goal together to avoid penalties for the entire population. As shown in left of Figure 6, CASEC with $\lambda_{\text{sp}} = 0.6$ fails to achieve learn the task properly. We load the best seed of CASEC with $\lambda_{\text{sp}} = 0.6$ and investigate what happens when it fails to solve the task. A typical failure case is visualized in the left side of Figure 8, that shows a partial slice of the `Gather` environment's map. Agents 1 and 2 need to join the rest of the group at g_1 to obtain the maximum reward of 10. In this scenario, CASEC fails to make such a transition. As a diagnostic, we visualize the agents overlain with the state-specific coordination graph and see that in this scenario the population indeed suffers from disconnectedness of the communication. Agent 1 cannot know what the rest of the agents are planning to do and thus it tries to achieve another goal by going to g_2 which causes the failure. When we manually force the sparsification to be done with ξ-metric as in KINGS, we observe that agent 1 is able to take the right action leading to the highest reward of 10, as shown on the right side of Figure 8 highlighting the core idea of our topologically informed sparsification.

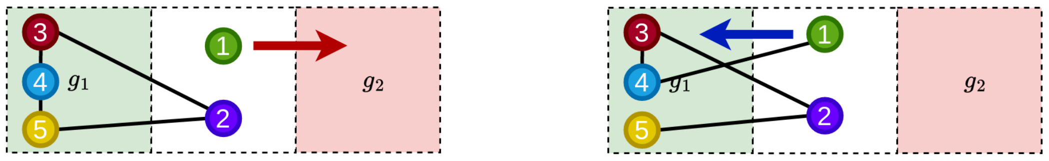

Figure 8: A visual example showing the importance of keeping the CG connected. (**Left**) CASEC disconnects agent 1 from the rest of the group, failing to obtain the optimal solution. (**Right**) ξ-sparsification employed by KINGS ensures that the group stays connected and arrives at g_1.

6 Conclusion

Our work enhances the mutual influence metric for deep Coordination Graph (CG) sparsification by incorporating cyclicity regularization. The method removes edges iteratively based on their mutual influence score while considering their significance for the entire graph's information flow. Integrating this metric into CG-based methods proves advantageous compared to topology-oblivious sparsification in many difficult coordination tasks. A notable limitation in current sparse coordination methods is the assumption of a static sparsity coefficient. Dynamically adjusting the number of active CG edges based on the state could offer a balanced solution, addressing max-sum constraints and optimizing the representational capacity of the joint utility model. Further research could explore more suitable distributed constraint optimization algorithms for CGs with high cyclicity.

Acknowledgments

This work was funded by the German Federal Ministry of Education and Research (BMBF) (Project: 01IS22078). The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project b187cb. NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG) – 440719683.

References

[1] Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *ICML*, 2020.

[2] Jacopo Castellini, Frans A Oliehoek, Rahul Savani, and Shimon Whiteson. The representational capacity of action-value networks for multi-agent reinforcement learning. *arXiv preprint arXiv:1902.07497*, 2019.

[3] Jesús Cerquides, Juan Antonio Rodríguez-Aguilar, Rémi Emonet, and Gauthier Picard. Solving highly cyclic distributed optimization problems without busting the bank: a decimation-based approach. *Logic Journal of the IGPL*, 29(1):72–95, 2021.

[4] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998.

[5] Liel Cohen and Roie Zivan. Max-sum revisited: The real power of damping. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Visionary Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pages 111–124. Springer, 2017.

[6] Alessandro Farinelli, Rogers Alex, Petcu Adrian, Jennings Nick, et al. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, volume 2, pages 639–646. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[7] Avinash Gautam and Sudeept Mohan. A review of research in multi-robot systems. In *2012 IEEE 7th international conference on industrial and information systems (ICIIS)*, pages 1–5. IEEE, 2012.

[8] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. *NIPS*, 2001.

[9] DJ Klein and O Ivanciuc. Graph cyclicity, excess conductance, and resistance deficit. *Journal of Mathematical Chemistry*, 30:271–287, 2001.

[10] Douglas J Klein. Resistance-distance sum rules. *Croatica chemica acta*, 75(2):633–649, 2002.

[11] Jelle R Kok and Nikos Vlassis. Sparse cooperative q-learning. In *ICML*, 2004.

[12] Victor Lesser, Charles L Ortiz Jr, and Milind Tambe. *Distributed sensor networks: A multiagent perspective*, volume 9. Springer Science & Business Media, 2003.

[13] Istvan Lukovits, S Nikolić, and N Trinajstić. Resistance distance in regular graphs. *International Journal of Quantum Chemistry*, 71(3):217–225, 1999.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[15] Andrea Montanari, Federico Ricci-Tersenghi, and Guilhem Semerjian. Solving constraint satisfaction problems through belief propagation-guided decimation. *arXiv preprint arXiv:0709.1667*, 2007.

[16] Conor Muldoon, Gregory MP O'Hare, Michael J O'Grady, Richard Tynan, and Niki Trigoni. Distributed constraint optimisation for resource limited sensor networks. *Science of Computer Programming*, 78(5):583–593, 2013.

[17] Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

[18] Afshin Oroojlooy and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, 53(11):13677–13722, 2023.

[19] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

[20] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, 2018.

[21] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10199–10210, 2020.

[22] Emma Rollon and Javier Larrosa. Decomposing utility functions in bounded max-sum for distributed constraint optimization. In *International conference on principles and practice of constraint programming*, pages 646–654. Springer, 2014.

[23] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[24] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *ICML*, 2019.

[25] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[26] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *ICML*, 1993.

[27] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.

[28] Jianhao Wang, Zhizhou Ren, Beining Han, Jianing Ye, and Chongjie Zhang. Towards understanding cooperative multi-agent q-learning with value factorization. *Advances in Neural Information Processing Systems*, 34:29142–29155, 2021.

[29] Tonghan Wang, Liang Zeng, Weijun Dong, Qianlan Yang, Yang Yu, and Chongjie Zhang. Context-aware sparse deep coordination graphs. *arXiv preprint arXiv:2106.02886*, 2021.

[30] Ermo Wei and Sean Luke. Lenient learning in independent-learner stochastic cooperative games. *The Journal of Machine Learning Research*, 17(1):2914–2955, 2016.

[31] Qianlan Yang, Weijun Dong, Zhizhou Ren, Jianhao Wang, Tonghan Wang, and Chongjie Zhang. Self-organized polynomial-time coordination graphs. In *International Conference on Machine Learning*, pages 24963–24979. PMLR, 2022.

[32] Yujun Yang. On a new cyclicity measure of graphs—the global cyclicity index. *Discrete Applied Mathematics*, 172:88–97, 2014.

A Motivating Example Description

For the motivating example discussed in Section 4.2, we choose the number of agents to be 11 and the action space to be of size 3. Then we take 1000 random seeds and for each one a fully connected CG induces a random joint utility function. We set individual utilities f_i to 0 and generate the payoff matrices f_{ij} by sampling values uniformly for each action pair (a_i, a_j) from the set $\{-1, 0, 1\}$. Finally we add noise to these sampled values from the standard normal distribution.

B Further Results on Changing Sparsity Coefficient

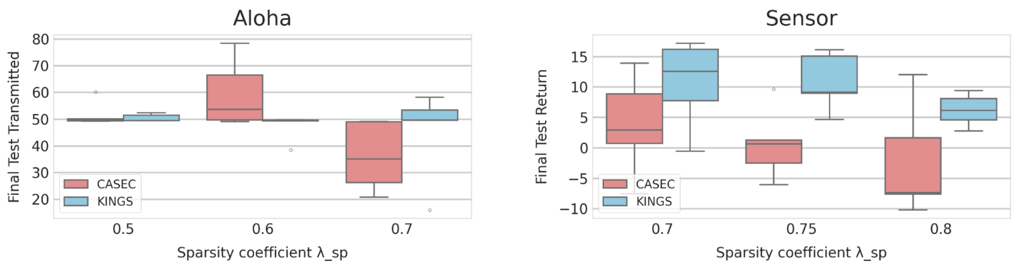

Figure 9: The effect of sparsity coefficient λ_{sp} on KINGS and CASEC. KINGS is in most cases able to achieve a higher performance for a fixed λ_{sp}.

As suggested originally by Wang et al. [29], `Aloha` and `Sensor` environments of the MACO benchmark are learned with very high levels of sparsity as shown in Table 1. In this part, we additionally test how the discussed sparsification methods behave with smaller values of λ_{sp} on these tasks. For `Aloha`, we plot the distribution of final performances for $\lambda_{\text{sp}} \in \{0.5, 0.6, 0.7\}$. For `Sensor` the corresponding set is $\{0.7, 0.75, 0.8\}$. The used sparsification budgets are 2 and 3 for `Aloha` and `Sensor`, respectively. Figure 9 presents the results. Generally, the final performance exhibits lower variance with KINGS, and especially in `Sensor`, it retains higher performance. The denser the graphs become, the less pronounced the difference between CASEC and KINGS is.

C Effect of ρ-formulation in MACO

Following Wang et al. [29], we formulate the payoff functions as a sum of a residual term ρ_{ij} and the individual utilities: $f_{ij} = f_i + f_j + \rho_{ij}$. In this setting, we are learning individual utilities and the residual term ρ_{ij} instead of directly learning payoff functions f_{ij}. While being a minor detail that is not discussed in the work that presents this architectural choice, it seems to have quite a strong effect on learning in MACO environments. Figure 10 presents illustrative results on this issue. The box plot demonstrates the final performance in `Sensor` for two different sparsity coefficients 0.7 and 0.75 when learning f_{ij} or via the residual formulation. Similar behaviour is observable for other MACO environments as well but we did not perform an extensive check on this. For fair comparison all the CG based approaches are implemented with this modification. Finally, due to computational constraints, we do not evaluate the significance of this choice on SMAC but instead adopt the choice learning f_{ij} directly as done by Wang et al. [29].

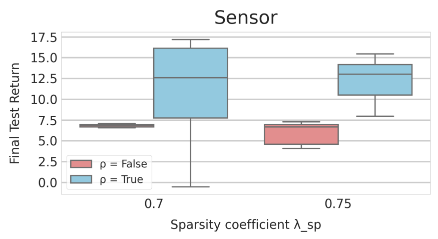

Figure 10: The effect of ρ-formulation.