Modular Value Function Factorization in Multi-Agent Reinforcement Learning

Oliver Järnefelt[1] and Carlo D'Eramo[2,3]

Abstract— **Real-world problems with multiple actors require them to coordinate while making decisions independently from each other. Typically, the large dimensionality and high unpredictability of the environment hinder the handcrafting or planning of effective behaviors. Multi-agent Reinforcement Learning (MARL) provides a framework for solving such problems by learning a parameterized policy for each agent that only depends on the state. Common approaches factorize the value functions of the agents enabling them to take independent decisions, or learn complex interactions by modeling the utility or payoff functions of the underlying coordination graph. In this paper, we discover the benefit of exploiting the connection between these two approaches. We propose to leverage the modularity of the embedded coordination graph by formulating the total utility as a sum of subteam mixings and prove that our modular factorization is able to cover the Independent-Global-Max (IGM) class of joint utility functions. We suggest finding the closest disjoint approximation of non-divisible graphs via graph partitioning, the quality of which we evaluate with a novel value-based partitioning distance measure. We derive theoretical and empirical advantages of our method evincing its benefit over baselines in several one-shot games, designed to highlight the promise of our modular factorization methods.**

I. INTRODUCTION

The ability to coordinate is a crucial requirement for autonomous systems, especially in real-world environments featuring multiple actors whose interactions have implications on the task dynamics. Consider for example the coordination requirements for an autonomous vehicle driving through a busy intersection or a warehouse robot assisting a human worker in moving heavy objects. As the complexity of the task increases, programming desired behaviors by hand becomes tedious and error-prone. In contrast, reinforcement learning (RL) [1] provides a flexible framework to learn optimal behavior from simple interactions with an environment, such that an agent observes the effect of its actions on the environment and optimizes its strategy. When transitioning from single-agent to multi-agent environments, the variety of problems significantly increases. As opposed to its single-agent counterpart, approaches in multi-agent reinforcement learning (MARL) [2] vary in their assumptions on, e.g., which communication constraints exist or how the reward provided by the environment is shared amongst the agents – whether it is shared globally or given locally to individuals, and if the latter is true, if the reward structure is zero- or general-sum which in turn determines the cooperativeness of the problem.

[1]Department of Computer Science, Aalto University, Finland oliver.jarnefelt@aalto.fi
[2] Department of Computer Science, TU Darmstadt, Germany
[3] Hessian.AI, The Hessian Center for Artificial Intelligence, Germany

In this work, we are interested in purely cooperative tasks – the joint behavior of the population determines a joint reward, which then gets evenly distributed among all of the agents. Furthermore, we require that the agents are unable to communicate explicitly during *deployment*. This implies that after the training phase, each agent needs to possess and adhere to a *learned protocol* that determines what to do in each situation. This widely-applied learning paradigm is known as *centralized training decentralized execution* (CTDE) [3]. A simple and natural solution is to train a population of agents independently, i.e., treat the other agents as part of the environment [2]. This approach is however hindered by the *multi-agent credit assignment* problem: given that the reward from the environment is shared equally between all agents, assessing the contribution of each individual becomes hard.

a) Related works: The introduction of deep neural networks to RL [4] has resulted in large advances in the field of cooperative MARL, leading to a surge of research activity around *value function factorization* (VFF) [5], [6]. VFF approaches train a centralized value function, which is some mixing of the individual agent utilities. 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. In parallel with VFFs, the *coordination graph* (CG) [7] has proven to be a powerful concept in modeling cooperative multiagent systems. CGs provide a graphical representation of the reward structure for a given state of the problem: the edges between agent nodes describe the factors that compose the overall value function. They are a suitable fit for describing many interesting applications – in internet routing each agent node has a routing table containing network topology information [8], in power grid optimization the agents communicate only to their neighbors determined by the grid layout [9] and in multi-agent control of individual robots whose embodiment consists of multiple limbs the CG structure is defined via the morphology specification [10]. The primary idea motivating our work is that in all of the previous cases, the influence of each agent has a limited scope within the population - depending on the CG topology big graphs can exhibit extensive *clustering* that implies invariances between subpopulations. Recent works have focused on investigating the representational capacities of CG-based value functions [11] and ways of scaling them up to enable efficient coordination in complex environments [12], [13]. A typical assumption made by CG-based methods is that agents of the population are able to communicate with the adjacent agents in the underlying coordination graph during execution – this is in contrast to our learning setting, where we assume agents to have no ability to share information to their neighbors when selecting actions.

b) Contribution: In this paper, we propose to leverage the formalism of CGs to provide structure to VFF approaches. In cases where the original problem is decomposable to smaller constituents which require little to no information exchange, we argue that learning should be *modular*, i.e., behavior for each constituent is learned via a local mixing function, which needs to only consider a subset of agent utilities to form the total value estimate. When this assumption holds, modular approaches can provide improvements in terms of learning speed. Flat approaches that do not exploit the locality of agent interactions have been shown to suffer from growing population sizes [14] – we expect methods utilizing the modularity principle to scale more favorably due to a more structured implicit credit assignment they provide to the actors. Simultaneously, compared to direct CG modeling approaches [11], [12], modular VFFs retain decentralizability and scale well even when the CG contains factors of more than 2 agents - the joint utility function has commonly been assumed to be a sum of pairwise factors to mitigate the combinatorial explosion in the number of model factors, an assumption our approach does not share. To this end, as our main contribution, we present two novel modular VFF algorithms, which we evaluate and analyze extensively in simple matrix games.

II. BACKGROUND

In this work, we consider one-shot multi-agent games which can be described as multi-agent extension of multi-armed bandits. A DEC-MDP [3] is summarized by a tuple

$$M := \langle \mathcal{S}, \mathcal{U}, \boldsymbol{d}, \boldsymbol{T}, \boldsymbol{r}, \gamma \rangle,$$

where the joint action space $\mathcal{U}$ is the concatenation of individual action spaces of all agents, and $\boldsymbol{d}, \boldsymbol{T}$ and $\boldsymbol{r}$ are multivariate extensions of their single-agent counterparts. For a given DEC-MDP, our objective is to find a mutually independent set of policies $\boldsymbol{\pi}^*$ that satisfies: $\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{u}\sim\boldsymbol{\pi}} Q^*_{\text{jt}}(s, \boldsymbol{u})$, where $Q^*_{\text{jt}}(s, \boldsymbol{u})$ denotes the total expected future discounted returns for the population. We refer to this quantity as the *optimal joint utility function*.

The basis of our work is QTRAN [6], which can be shown to cover the entire Independent-Global-Max (IGM) condition satisfying joint utilities, i.e., the ones for which $\bar{\boldsymbol{u}} = \arg\max_{\boldsymbol{u}} Q_{\text{jt}}(s, \boldsymbol{u}) = \{\arg\max_{u_i} Q_i(s, u_i)\}_{i=1}^N$ holds. To do this, QTRAN ties the independent actor utilities Q_i to the total utility Q_{jt} via the following connection:

$$\sum_{i\in\mathcal{I}} Q_i(s, u_i) - Q_{\text{jt}}(s, \boldsymbol{u}) + V_{\text{jt}}(s) = \begin{cases} 0 & \boldsymbol{u} = \bar{\boldsymbol{u}} \\ \geq 0 & \boldsymbol{u} \neq \bar{\boldsymbol{u}} \end{cases}$$

where $V_{\text{jt}}(s) = \max_{\boldsymbol{u}} Q_{\text{jt}}(s, \boldsymbol{u}) - \sum_{i=1}^N Q_i(s, u_i)$. In practice, L2 relaxations of the constraints are instead enforced via a temporal difference loss $\mathcal{L}_{TD}$ and a matching loss, which is weighted sum of two terms $\mathcal{L}_{nopt}$ and $\mathcal{L}_{opt}$.

III. METHOD

The key observation for our work is that there exists many practical scenarios where 1) the underlying coordination graph structure is either available by construction or can be built without incurring large computational burden and 2) the structure exhibits a level of *modularity*. By the latter we mean that there are subpopulations of agents whose behaviors are mutually independent eliminating the need of communication between the groups. Depending on the *exactness* of the modularity, we propose two different approaches to leverage it as described in the following subsections.

A. Known Exact Structure Leveraging

We are interested in problems whose $Q^*_{\text{jt}}(s, u)$ adheres to the IGM condition and embeds by a Disconnected Hyper Coordination Graph (DHCG), which is a particular subclass of CGs [7].

Definition III.1 (Disconnected Hyper Coordination Graph)**.** *A DHCG $\mathcal{H}$ is disjoint set of K components or partitions, which themselves are hypergraphs*[1]*. Denote $\mathcal{V}$ as the total set of all agents/vertices, and $\mathcal{E}$ as the total hyperedge set of the graph. Let each component $\mathcal{C}_i := \langle \mathcal{V}_i, \mathcal{E}_i \rangle$, where $\mathcal{V}_i$ denotes the agents/vertices in component $\mathcal{C}_i$ and $\mathcal{E}_i$ the set of hyperedges in $\mathcal{C}_i$. Each hyperedge $e \in \mathcal{E}_i$ connects $K \geq 1$ vertices in $\mathcal{V}_i$ and corresponds to factor q_e of the total utility function. For the component vertices it holds that: $\mathcal{V} = \bigcup_{i=1}^K \mathcal{V}_i$ and $\mathcal{E} = \bigcup_{i=1}^K \mathcal{E}_i$. For disconnectivity, we further require that $\mathcal{V}_i \cap \mathcal{V}_j \neq \emptyset \iff i = j$ and $\mathcal{E}_i \cap \mathcal{E}_j \neq \emptyset \iff i = j$. Then DHCG is given as the union of the disjoint components: $\mathcal{H} = \bigcup_{i=1}^C \mathcal{C}_i$. The total joint utility function is the sum over all hyperedge factors: $Q_{jt}(s, \boldsymbol{u}) = \sum_{e\in\mathcal{E}} q_e(s, \boldsymbol{u}_e)$, where $\boldsymbol{u}_e$ denotes that actions selected by the agents belonging to e.*

We propose the Known Exact Structure Leveraging (KESL) method to exploit the independence between DHCG components by learning each one with a separate QTRAN mixing. The total utility trained with the TD error is:

$$Q_{\text{jt}}(s, \boldsymbol{u}) = \sum_{j=1}^K Q^{(j)}_{\text{jt}}(s, \boldsymbol{u} \cdot \mathcal{K}(j; s)),$$

where $Q^{(j)}_{\text{jt}}$ is the estimated joint utility for component $\mathcal{C}_j$, and $\mathcal{K}(j; s) \in \{0, 1\}^{N\times K}$ is a component assignment matrix, where each row i is a one-hot vector where the index of 1 tells which to component agent i gets assigned to. To verify that KESL is compatible with the CTDE setting, we present the following proposition (proof is omitted for brevity[2]):

Proposition 1. *Let $\{Q_i\}_{i=1}^N$ denote the utility functions for each agent and Q_{jt} the joint utility computed by KESL, with IGM guaranteeing component mixers $\{Q^{(j)}_{jt}\}_{j=1}^K$ and any*

[1]Generalization of a normal graph with the difference that now edges can connect more than 2 vertices.

[2]We still note that it is very similar to that of proof of Theorem 1 in [14]

component assigner $\mathcal{K}$. For these quantities it holds that

$$\arg\max_{\boldsymbol{u}} Q_{jt}(s, \boldsymbol{u}) = \{\arg\max_{u_i} Q_i(s, u_i)\}_{i=1}^{N}.$$

B. Known Approximate Structure Leveraging

With *non-divisible* CGs of large population sizes, in which communication is mostly *partitioned* and inter-partition communication happens only through small bottlenecks, we interpret the partitions as approximately independent components and learn to mix each one. We propose to find the closest disjoint partitioning $\mathcal{P}$ of the original graph. After partition assignment, we apply the steps of KESL on the partitions to form joint utility. We name this process Known Approximate Structure Leveraging (KASL).

While we leave the practical implementation of finding the closest disjoint partitioning for future work, we still take the first step to evaluating the quality of partitionings. To this end, we propose a novel partitioning distance measure that takes into account the functional form of the factors that get broken by the partitioning.

Definition III.2. *Consider a hypergraph $\mathcal{H} = \langle \mathcal{V}, \mathcal{E} \rangle$, and partitioning $\mathcal{P}$. We define the partitioning distance $d(\mathcal{H}, \mathcal{P})$ as the minimum estimation error achievable by any joint utility function in the family of factorizations determined by $\mathcal{P}$:*

$$d(\mathcal{H}, \mathcal{P}) = \min_{\mathcal{H}' \in \mathcal{P}} ||Q^*_{jt} - Q^*_{jt,\mathcal{H}'}||_\infty \tag{1}$$

Proposition 2. *Given a hypergraph $\mathcal{H} = \langle \mathcal{V}, \mathcal{E} \rangle$, let $\mathcal{E}_c \subseteq \mathcal{E}$ denote the set of cut hyperedges*[3] *by partitioning $\mathcal{P}$ that introduces new hyperedges $\mathcal{E}_n = \{e' \mid \exists\ e \in \mathcal{E}_c \text{ s.t. } e' \subset e\}$. Further, given a hyperedge pair $\langle e, e' \rangle$, with $e \in \mathcal{E}_n$ and $e \in \mathcal{E}_c$, we denote $\mathcal{V}_+ = \{v \mid v \in e',\ e' \subset e\}$ and $\mathcal{V}_- = \{v \mid v \in e',\ e \setminus e'\}$ and $\tilde{q}_{e,+} = \frac{1}{|\mathcal{U}_-|} \sum_{\boldsymbol{u}'_- \in \mathcal{U}_-} q_e(\cdot, \boldsymbol{u}'_-)$. Then, the minimum estimation error achievable by any utility function within the family of factorizations determined by $\mathcal{P}$ is upper bounded by (proof omitted for space limitations):*

$$\min_{\mathcal{H}' \in \mathcal{P}'} ||Q^*_{jt} - Q^*_{jt,\mathcal{H}'}||_\infty \leq \sum_{e \in \mathcal{E}_c} \sum_{e' \in \mathcal{E}_n, e' \subset e} ||q_e - \tilde{q}_{e,+}||_\infty.$$

IV. EXPERIMENTS

To understand the strengths and weaknesses of our approaches, we compare them with a set of carefully chosen baselines in two matrix games, Deadliest Catch (DC) and Generalized Firefighting (GFF), described in the following subsections. To measure the benefit of breaking the original problem into multiple smaller ones according to the DHCG components, we compare against QTRAN [6] trained on the entire population. While QTRAN should be able to learn all the tasks, we expect to see it slowing down as N grows. The QTRAN constraints imposed to ensure IGM property are not exactly enforced but instead a soft relaxation of the original problem is solved; when N grows, so does the number of relaxed optimization constraints, thus, possibly hurting accuracy of the factorization. We analyze where our method stands w.r.t. to state-of-the-art coordination graph modeling methods, by testing the performance of two versions of DCG [12] – with (DCG) and without (CG[4]) parameter sharing between learned coordination graph factors. Finally, as a simple but naive approach for the tasks discussed above, we also test the Independent Q-Learners (IQL) [2].

In all our experiments, we use ϵ-greedy policy, with $\epsilon = 1$ for data gathering, which we do for $20,000$ and $5,000$ steps for DC and GFF, respectively. Each method is run for 10 seeds for every environment. For evaluation, we use $\epsilon = 0$. The shaded areas in all the plots denote the 95% confidence interval. The agent networks as well as the Q_{jt} and V_{jt} of every QTRAN mixer are all ReLU activated fully-connected neural networks with two hidden layers and hidden dimensionality 32. We use RMSProp with learning rate $5e-4$, $\alpha = 0.99$ and $\epsilon = 0.00001$, with gradient clip with max $\ell_2 = 10$, as suggested by e.g. [12]. For QTRAN/KESL/KASL learning, we use $\lambda_{\text{opt}} = \lambda_{\text{nopt}} = 1$. All our implementations are based the PyMARL [15] framework. For CG and DCG, we used the original implementations from the authors [5].

A. Deadliest Catch

We begin the experimental section by studying the most favorable setting to our method, i.e., when the ground truth CG is actually modular. For this purpose, we introduce Deadliest Catch (DC), a simple matrix game in which a fishing company dispatches K fleets to K island groups. Each island group hosts $|\mathcal{U}|$ known fishing hotspots and each vessel of a fleet needs to make a choice on which fishing spot u it goes to. Each island group has their individual hotspot $\bar{\boldsymbol{u}}$ which hosts a giant squid that needs to be fished by all vessels of the fleet to succeed, otherwise the squid will destroy all the boats trying to catch it incurring penalty of -5 units. Company can sell each squid for 5 units and the smaller catches from other fishing spots for with 1 unit. For each fleet $\mathcal{F}$ the reward is thus:

$$r_i = \begin{cases} 5 & \text{if } \bar{\boldsymbol{u}}_{\mathcal{F}} = u_j\ \forall j \in \mathcal{F} \\ -5 & \text{if } \exists j \in \mathcal{F},\ \bar{\boldsymbol{u}}_{\mathcal{F}} \neq u_j \\ 1 & \text{otherwise} \end{cases}$$

resulting in the joint reward of $r_{\text{jt}} = \sum_{i=1}^{K} r_i$.

Game-theoretically, each component plays the classical multi-agent generalization of the stag-hunt game with shared rewards, also called the climbing game [16]. Each sub-game of the proposed environment is testing the algorithms capability to avoid relative overgeneralization, and combining multiple parallel sub-games for the total reward exacerbates the issue. To test the effect of varying the number of vertices or the sizes of each independent component in the underlying

[3]An edge is said to be cut if a partitioning assigns vertices of the edge to different partitions.

[4]We thus overload the CG abbrevation - when there is a chance for confusion, CG refers to the method.

[5]https://github.com/wendelinboehmer/dcg.

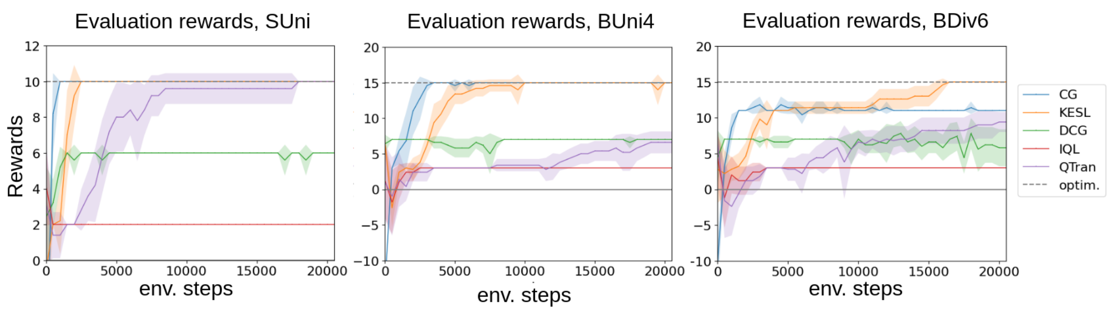

Fig. 1: Performances on DC environments

coordination graph, we benchmark the relevant methods on 3 instances of DC, whose parameters are specified in Table I. Plots in Figure 1 demonstrate the results in all of the three

Name	# agents	Fleet sizes	$\bar{u}$ for each fleet
Small-Uniform (SUni)	6	{3,3}	[0,1]
Big-Uniform4 (BUni4)	12	{4,4,4}	[0,1,2]
Big-Diverse6 (BDiv6)	12	{3,3,6}	[0,1,2]

TABLE I: Specifications of the DC environments tested.

environments. A few key observations can be made:

1) QTRAN performance degrades when increasing the population size from $N = 6$ (SUni) to $N = 12$ (BUni4 & BDiv6). We investigate this in the end of this subsection;
2) DCG gets stuck to a performance 5-6 in every environment; the parameter sharing between learned payoff functions employed by the approach leads to similar action selection between agents, as demonstrated in (c) and (d) matrices of Figure 2. As a result, DCG is prone to getting stuck to behavior where only some components of the DHCG are learned;
3) CG performs strongly on each of the tasks, but seems to converge prematurely in BDiv6. Whereas it is not hindered by parameter sharing, we hypothesize that since it is modeling the total value function as a sum of pairwise interactions, components that require high number of agents to solve should be hard to learn – convincingly, in this case it is indeed the 6-agent component that does not get learned, as can be seen in (b) matrix of Figure 2;
4) IQL demonstrates weak behavior in each game, due to its inability to overcome relative overgeneralization;
5) KESL solves each of the environments in $20,000$ steps. While, it is not immune to the increase in N or the component sizes, it seems to provide significant boost to flat QTRAN while still having the ability to large components which prove hard for even CG.

As demonstrated by Figure 1, the performance gap between QTRAN and KESL is significant. We investigate the source of this difference by studying the structure of the learning signal propagating back to individual agents for both the flat and modular approaches in BUni4. The learning signal

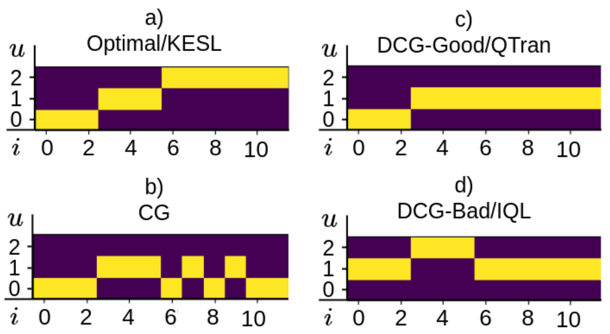

Fig. 2: A typical final greedy policies in BDiv6 for all of the methods. The x-coordinate is the agent id i, and the y-coordinate the action id u. Agents 0-2 belong to fleet 0, agents 3-5 to fleet 1 and agents 6-11 to fleet 2.

flowing back to individual agents i in QTRAN/KESL is:

$$g_i = \frac{\partial(\mathcal{L}_{\text{nopt}} + \mathcal{L}_{\text{opt}})}{\partial Q_i}.$$

Since the environment requires agents within one component to take same actions, and the actions need to be different across different components to achieve the maximum reward, successful methods must have similar gradients within group but dissimilar across groups. To quantify the clustering, we first cluster the gradients of each agent with k-means [17], after which we compute the normalized mutual information of predicted cluster labels:

$$\text{NMI}(Y;C) = \frac{\mathcal{I}(Y;C)}{\mathcal{H}(Y) + \mathcal{H}(C)},$$

where $\mathcal{I}$ denotes the Shannon mutual information, $\mathcal{H}$ the Shannon entropy, Y the ground truth group/graph component labels, and C the predicted cluster labels. Intuitively, NMI quantifies the amount of information that is gained about class label Y given the knowledge of the cluster label C, and we expect it to be higher for modular methods as they explicitly consider the independencies between task components. The intuition of gradients exhibiting task related structural information is however piggybacking on the assumption that the matching loss $\mathcal{L}_{nopt} + \mathcal{L}_{opt}$, has an accurate target Q_{jt} – thus we also compute the TD loss of the estimated joint utility function to assess the stability of the learning signal flowing back to the agents. As can be seen from Figure 3, KESL structuring improves the convergence of the joint utility function noticeably, thus providing more stable targets for the decomposable joint utility function Q'_{jt}. This leads KESL agents to obtain more structured gradients resulting in higher values for NMI of KESL in Figure 4, which translates into better evaluation performance compared to QTRAN, as shown in the center plot of Figure 1.

In conclusion, the experiments conducted in the Deadliest Catch environments demonstrate that VFF approaches suffer dramatically from the degradation of sample-efficiency as the number of participating agents increases; if the task structure allows for learning component-wise mixing, KESL can significantly boost the learning speed over flat methods. The improvements in evaluation performance KESL demonstrates over QTRAN are attributable to more sample-efficient estimation of Q_{jt} and more appropriately structured learning signal. Furthermore, the performance of methods estimating the underlying coordination graph by learning pairwise factors can deteriorate substantially the cardinality of the underlying graph components increases. KESL provides robustness against this kind of variation due to its lack of assumptions on the edge cardinality.

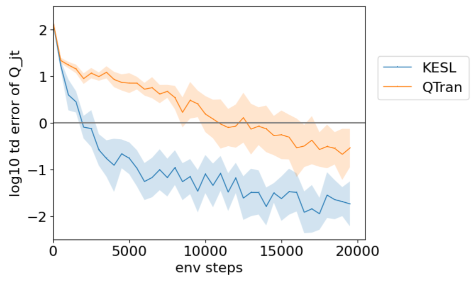

Fig. 3: Log10 of the TD error for Q_{jt}.

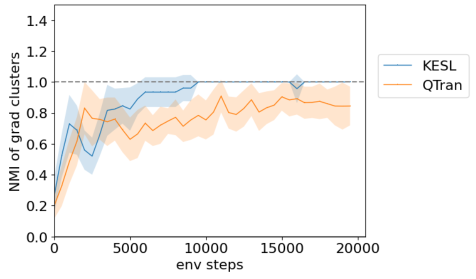

Fig. 4: NMI for the gradient clusters.

Agent	Precinct	Reward Factors
0	{0,1,2,3}	{3}
1	{3,4,5,6}	{3}
2	{4,7,8,9}	{3,9}
3	{9,10,11,12}	{9,12}
4	{12,13,14,15}	{12}
5	{12,16,17,18}	{12}

TABLE II: Precincts and reward factors agents belong to.

B. *Generalized Firefighting*

We benchmark instances of KASL against QTRAN in a modified version of the Generalized Firefighting game (GFF) [18] to study if approximate modularity can be utilized when the original graph is not divisible into separate components, as suggested in section III-B. The game consists of $N = 6$ firefighters fighting fires in $|\mathcal{U}| = 4$ houses. In total there are 19 houses labeled with indices 0-18; only the houses 3, 9 and 12 are on fire. In contrast to the original GFF, we assume that the fire in house number 9 is more dangerous than the other two: extinguishing it requires at least 2 agents. The modification is motivated by our interest in showing how our proposed partitioning distance works with CGs whose factors vary in their importance to the total utility. Let #("house i") be the number of agents chosen to extinguish the fire at house i; the reward associated with house 9 is

$$r_9 = \begin{cases} 2 & \text{if } \#(\text{"house 9"}) = 2 \\ -2 & \text{if } \#(\text{"house 9"}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

For the other two burning buildings, reward is sub-additive:

$$r_i = \min\{\#(i), 1\} + \max\{\#(i) - 1, 0\}/2, \;\; i \in [3, 12].$$

Table II shows the house numbers each agent is responsible for, and which reward factor each agent belongs to. The left

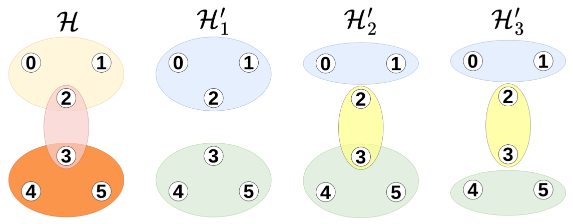

Fig. 5: The factors for the burning buildings of the original coordination graph $\mathcal{H}$, q^*_{012} (house 3), q^*_{23} (house 9) and q^*_{345} (house 12) are shown on the left. $\mathcal{H}'_1$, $\mathcal{H}'_2$ and $\mathcal{H}'_3$ are the optimal factorizations for three different partitionings of $\mathcal{H}$: $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$, respectively.

graph in Figure 5 shows the ground truth CG $\mathcal{H}$ of the task. $\mathcal{H}'_1$, $\mathcal{H}'_2$ and $\mathcal{H}'_3$ are the best factorizations for partitionings $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$, respectively. Using the optimal factorizations, the partitioning distances as defined in section III-B are:

$$\begin{aligned} d(\mathcal{H}, \mathcal{P}_1) &\leq ||q^*_{23} - q'_{\emptyset}||_\infty = 2 \\ d(\mathcal{H}, \mathcal{P}_2) &\leq ||q^*_{012} - q'_{01}||_\infty = 0.75 \\ d(\mathcal{H}, \mathcal{P}_3) &\leq ||q^*_{012} - q'_{01}||_\infty + ||q^*_{345} - q'_{45}||_\infty = 1.5, \end{aligned}$$

where $\emptyset$ denotes an empty set: given that q^*_{23} cannot be estimated by the best factorization of $\mathcal{P}_1$, the factor error is computed by finding the infinity norm of the cut edge. These distances reflect the intuition that, as the reward component corresponding to house 9 has a large variance and magnitude relative to houses 3 and 12, $\mathcal{P}_1$, that is not able to represent the factor q_{23}, should induce the largest error amongst the partitionings.

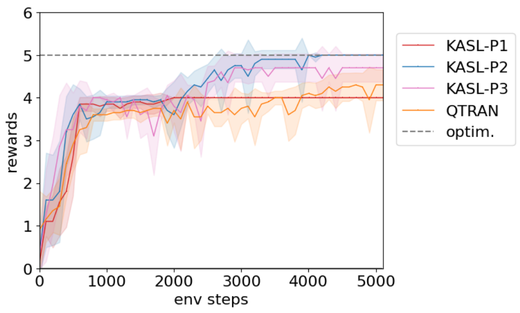

Fig. 6: Firefighting results for KASL with partitionings $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ and QTRAN.

Convincingly, the distance measure is aligned with the empirical performance achieved by KASL – the lower it is, the higher the final evaluation performance is. Figure 6 shows the performance plots for QTRAN and KASL with each of the partitionings. KASL with $\mathcal{P}_1$ gets stuck into suboptimal behavior shown in Figure 7. This highlights the issue of cutting the important q_{23} edge in $\mathcal{H}$, as the learning processes of agents 2 and 3 are separated, they cannot learn to distinguish between coordinative and uncoordinative actions and thus experience relative overgeneralization and learn to fight fires in buildings 3 and 12, respectively. Surprisingly, QTRAN gets stuck to suboptimal behavior similar to that of KASL-$\mathcal{P}_1$. We suspect that the reason for this is simply the learning speed: given that KASL is able to learn on more compact set of agents, it is faster to learn the optimal policy, provided a consistent value function to an optimal policy is representable by the employed partitioning.

To summarize: in GFF, we demonstrate that the quality of the applied partitioning influences the empirical performance of the algorithm – important factors, that are highly impactful to the total task return, must be represented accurately to learn optimal decentralized policies. Furthermore, these tests provide encouraging results implying that despite the non-divisibility of the original CG, we can find useful disjoint approximations using which we can boost VFF learning.

V. CONCLUSIONS AND FUTURE WORK

We have shown that injecting CG information to the value function factorization by learning component-wise mixing is helpful to increase sample-efficiency, providing stabler learning signal compared to flat approaches. In cases where underlying CG is not decomposable but still contains approximately separable partitions, we apply modularized VFF on the partitioned graph retaining high performance. While this work focuses on the evaluation of several competing partitionings, we will extend the idea of the partitioning distance metric to finding optimal partitionings. Such method opens the door for deploying KASL to realistic multi-agent environments where flat approaches are unusable.

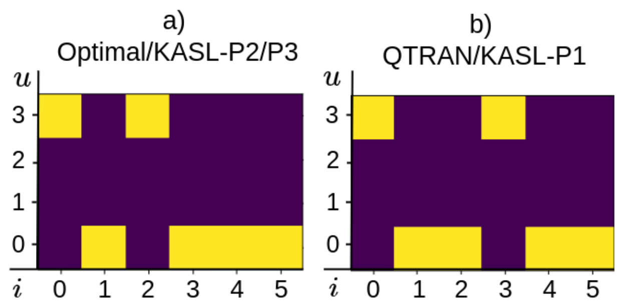

Fig. 7: The x-coordinate is the agent id i, and the y-coordinate the action id u - agent i taking u-th action corresponds to agent i going to the u-th house in their precinct specified in Table II. (Left) Optimal/typical behaviors of KASL-$\mathcal{P}_2$/$\mathcal{P}_3$, (Right) Common suboptimal behavior demonstrated by QTRAN/KASL-$\mathcal{P}_1$; agent 2 takes action 0 (goes to house 3) and agent 3 takes action 3 (goes to house 12), which shows that the methods avoid the house 9 that needs to be extinguished for the highest reward.

ACKNOWLEDGMENT

Funded by the German Federal Ministry of Education and Research (BMBF) (Project: 01IS22078) and the project 'The Third Wave of Artificial Intelligence – 3AI' funded by the Ministry for Science and Arts of the state of Hessen.

REFERENCES

[1] R. S. Sutton *et al.*, *Reinforcement learning: An introduction*. MIT press, 2018.
[2] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *ICML*, 1993.
[3] F. A. Oliehoek *et al.*, "Optimal and approximate q-value functions for decentralized pomdps," *JAIR*, 2008.
[4] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[5] P. Sunehag *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.
[6] K. Son *et al.*, "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *ICML*, 2019.
[7] C. Guestrin *et al.*, "Multiagent planning with factored mdps," *NIPS*, 2001.
[8] J. Boyan *et al.*, "Packet routing in dynamically changing networks: A reinforcement learning approach," *NIPS*, 1993.
[9] D. Chen *et al.*, "Powernet: Multi-agent deep reinforcement learning for scalable powergrid control," *IEEE Transactions on Power Systems*, 2021.
[10] W. Huang *et al.*, "One policy to control them all: Shared modular policies for agent-agnostic control," in *ICML*, 2020.
[11] J. Castellini *et al.*, "The representational capacity of action-value networks for multi-agent reinforcement learning," *arXiv preprint arXiv:1902.07497*, 2019.
[12] W. Böhmer *et al.*, "Deep coordination graphs," in *ICML*, 2020.
[13] T. Wang *et al.*, "Context-aware sparse deep coordination graphs," *arXiv preprint arXiv:2106.02886*, 2021.
[14] T. Phan *et al.*, "Vast: Value function factorization with variable agent sub-teams," *NIPS*, 2021.
[15] M. Samvelyan *et al.*, "The starcraft multi-agent challenge," *arXiv preprint arXiv:1902.04043*, 2019.
[16] C. Claus *et al.*, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, 1998.
[17] H. Steinhaus *et al.*, "Sur la division des corps matériels en parties," *Bull. Acad. Polon. Sci*, 1956.
[18] F. Oliehoek, *Value-based planning for teams of agents in stochastic partially observable environments*. Amsterdam University Press, 2010.