The challenge of redundancy on multi-agent value factorisation

Siddarth Singh, Benjamin Rosman
School of Computer Science and Applied Mathematics
The University of the Witwatersrand
Johannesburg, South Africa
siddarthsingh2@gmail.com, benjamin.rosman1@wits.ac.za

Abstract

Recently there has been great development in the field of multi-agent reinforcement learning (MARL). In the cooperative partially observable multi-agent setting central value functions have been used to perform multi-agent credit assignment for joint global rewards. The standard solution is the use of centralised training and decentralised execution where a central critic conditions the polices of the cooperative agents based on a central observation. Simulated training environments designed using video games typically only contain exactly the number of agents required to solve the tasks based on preexisting knowledge of the game dynamics and human player solutions. In a more general case, there is likely to be a larger number of agents in an environment than is required to solve the task. These redundant agents reduce overall performance by enlarging the dimensionality of the ground truth state if available and increasing the size of the joint policy used to solve the environment. In the case where no ground truth state is available a concatenation of all local observations is used which scales in size with the number of agents and becomes insufficient to condition the centralised critic in large spaces. We propose leveraging layerwise relevance propagation (LRP) to instead separate the learning of the joint value function and generation of local reward signals and create a new MARL algorithm: relevance decomposition network (RDN). We compare our method to other state of the art MARL algorithms in challenging StarCraft2 and simpler matrix game environments. We show that decomposition algorithms' performance and the usefulness of the state space degrades as the number of redundant agents increases.

1 Introduction

For most multi-agent tasks in a practical setting we would not know the precise number of agents required to optimally solve the problem. In a more general case, there is likely to be a larger number of agents in an environment than is required to solve the task. As the number of independent agents increases so would the size of a ground truth state space that most problems require. Also, as the state space increased it would become increasingly difficult to interpret.

In complex environments constructing an accurate ground truth representation becomes difficult. An example of this is the Piano Movers problem [1] when applied to the robotics setting. The problem can be described as follows: given a body B and a region bounded by a collection of walls, either find a continuous motion connecting the two given positions and orientations of B during which B avoids collisions with walls, or else establish that no such motion exists. When applying a solution to this task in real life we are limited by the data obtained from sensors and cameras on the robots in use which is typically noisy and of limited accuracy. This data is the local observations of the independent agents and is an egocentric local partial observation of the environment from the perspective of each agent. A concatenation of these local observations is still only a partial representation of the full state space and in the centralised training decentralised execution (CTDE) paradigm using this results in sub-optimal performance for most algorithms [2]. Given the limitation of the observable space in real or realistic settings it is not reliable to assume that an easily interpreted state approximation can be made from only these limited local observations. When there is a large number of agents many of the agents in the group can be redundant for achieving and optimal policy. These redundant agents increase the previous state space interpretation issue as the ground truth space usually increases to represent the space of these additional agents. Therefore it is important to develop algorithms that can effectively separate agents that are essential to solve a task and agents that are redundant to allow MARL algorithms to be deployed into more realistic and eventually real world challenges.

We consider the idea of a collaborative task with a small margin of error, like the traditional Piano Movers problem. However we consider the case where only n of m total agents in the environment are required to effectively complete the task. In the Piano Movers problem consider the case where an arbitrarily large number of agents are required to re-position the piano with each agent occupying a fixed amount of space in the environment along with the piano. Realistically with a large number of agents only a small subgroup n of$m$ will be required to solve the task under the joint optimal policy. Ultimately during training the policy will change to minimise the effect of certain agents on the overall outcome as the joint optimal policy only required that they do not act in a manner that is destructive to the actions of the smaller group of required agents.

In cooperative MARL the most common methods currently make use of the CTDE paradigm [3]. A centralised critic is used which has access to the central state and is used to condition the local policies of the agents based on their local observations. A flaw of this method is the reliance on an easy to interpret and extract ground truth state. When considering complex or high dimensional environments this may be unavailable or intractable.

In the Qatten [4] paper the idea is put forward that in both the linear and monotonic case value factorisation algorithms exhibit poor performance in environments with a high number of redundant agents as it it difficult to assign credit for task completion accurately when a disproportionately high credit is assigned to only a small number of the total agent set. They come to this conclusion by analysing the attention weights of their attention based critic network which showed in order to solve this environment a high reward needed to be assigned to a small subset of the agents which are the most relevant towards the success of the joint policy.

In this paper we propose the method relevance decomposition network (RDN) which makes use of layerwise relevance propagation (LRP) [5] as an alternative to learned value decomposition only using local agent observations. By not using learned decomposition we can separate the learning of the joint value function from the training of the independent agents. This allows us to make use of existing methods in the independent deep reinforcement learning (DRL) domain to enhance the performance of the agents. Additionally we are able to make full use of the relationships between the local observations of the agents, their identities and their actions at each timestep to decompose the relationship between the global and local rewards.

We also perform ablative studies to show how abusing the full observation space can inflate the scores of the learned policies. We use LRP to perform multi-agent credit assignment in a cooperative MARL environment. Rather than condition the agents' polices on the central state during training time we directly calculate the contributions of the observations, actions and identities of each agent towards the joint value function. In section 2 we cover the background information required for the paper including decentralised POMDPS (Dec-POMDPS) and LRP, section 3 presents our method RDN, section 4 covers our experiment methodology and introduces our baseline algorithms, training environments and architecture choices, section 5 contains the experiment results and interpretations, section 6 contains the related works and finally section 7 concludes and summarises the findings of the paper.

2 Background

2.1 Dec-POMDPs

Fully cooperative multi-agent tasks can be modelled as Dec-POMDPs [6]. Formally a Dec-POMDP G is given by a tuple:

$$G = \langle S, U, P, r, Z, O, n, \gamma \rangle \tag{1}$$

Where $s \in S$ describes the true state of the environment. At each timestep, each agent $a \in A \equiv \{1, ..., n\}$ chooses and action $u^a \in U$, forming a joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$. This causes a transition of the environment according to the state transition function $P : S \times \mathbf{U} \to [0, 1]$. All agents share the same reward function $r(s, \mathbf{u}) : S \times \mathbf{U} \to R$ and $\gamma \in [0, 1)$ is a discount factor.

Each agent draws individual observations $z \in Z$ according to observation function $O(s, a) : S \times A \to Z$. Each agent has an observation history $\tau^a \in T \equiv (Z \times U)^*$ on which it conditions a stochastic policy $\pi^a(u^a|\tau^a) : T \times U \to [0, 1]$. The joint policy π has a joint *action-value function* : $Q^\pi(s_t, \mathbf{u}_t) = E_{s_{t+1:\infty}, \mathbf{u}_{t+1:\infty}}[R_t|s_t, \mathbf{u}_t]$, where $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the *discounted return.*

In order to solve Dec-POMDPs we must find a *joint policy*. This is a set of polices mapped to individual agents. Additionally as agents are not able to directly observe the global state it is a requirement to learn a history of observations. Using this history an agent is able to map a local policy to its local POMDP. A local policy refers to the policy used by an individual agent based on the limited local observation it obtains from the environment. The local POMDP refers to the Partially Observable Markov Decision Process used in the decision making process for an individual agent based on its local observations from the environment and the actions its policy produces. Unlike single agent systems we cannot usually estimate the hidden state from individual agent observation histories as it is no longer stationary and becomes dependent on the behaviours of all agents in the joint system [7].

2.2 Layerwise-Relevance Propagation

Layerwise-Relevance Propagation (LRP) is a method that has been developed to interpret neural network based machine learning models [5]. It assigns a relevance or importance value to each node of the neural network (NN) which describes how much it contributed to the final network output. The relevance values are calculated recursively from the output to the input layer of the NN.

The propagation of values by LRP is subject to a conservative property, where the total relevance assigned to a neuron is equal to the total relevance assigned to the neurons in the previous layer of the NN . Formally let j and k be two consecutive layers of the neural network. Propagating relevance scores R_k at a given layer onto neurons of the previous layer is achieved by applying the rule:

$$R_j = \sum_k \frac{z_{jk}}{\sum_j z_{jk}} R_k. \tag{2}$$

Where z_{jk} models the extent to which neuron j has contributed to make the neuron k relevant with $z_k = \epsilon + \sum_{0,j} a_j \cdot \rho(w_{jk})$ where ϵ is a small constant to aid numerical stability, ρ is an arbitrary function applied to the weights of the neurons in the NN layer and a is the output from neuron j in the earlier layer of the NN to neuron k. Using equation 2 above for all neurons in a network we can verify the layer-wise conservation property $R_j = \sum_k R_k$ where R_j is a vector of relevance scores with each neuron $k's$ relevance scores indexed as R_{jk}, and by extension the global conservative property $\sum_k R_k = f(x)$ where $f(x)$ is the output of the NN for some input x.

2.3 Related Work

Value decomposition network (VDN) [8] introduces the first central value decomposition framework for the multi-agent cooperative setting in Deep Reinforcement Learning (DRL). In this framework we receive only the global reward which we must then learn to distribute across all independent agents based on their contributions. This framework makes the assumption of an additive relationship between agent rewards $Q_{tot} \approx \sum_{i=1}^{n} Q_i(o^i, a^i)$ where Q_{tot} is the global reward at the current step and Q_i is the individual rewards for agent i. Essentially the sum of the rewards of each individual agent is approximately the same as the global reward at any timestep. The assumption of a linear relationship has been shown to be restrictive and leads to the formation of sub optimal policies even in simple gridworlds and again in more difficult scenarios in SMAC. In particular it struggles in the *bane_no_z* environment where there is a large number of redundant agents.

Building on the framework of VDN is **QMIX** [9] which relaxes the additive assumption by forcing the evaluation of a monotonic function through the use of hypernetworks. This monotonic constraint allows a more relaxed formation of the relationship between Q_{tot} and each individual Q_a. To ensure monotonicity we use the constraint $\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in A$ where ∂Q_{tot} the gradient of the global reward and ∂Q_a the gradient of each individual agents' rewards are both monotonic. The weights of the central value function (CVF) which is the central critic used to factorize the global rewards are produced using hypernetworks which produce a vector which is then reshaped to the appropriate size. Hyper networks are neural networks that are used to learn the weights of another neural network. Each hyper network uses absolute activation functions so that the weights of the CVF are always non-negative. A limitation of this method is that it is only able to decompose monotonic reward functions which limits the range of useful environments and to achieve maximum performance on complex environments it requires the use of a clear representation of the ground truth global state to accurately decompose the reward function. Much like VDN it also struggles in the case of high numbers of redundant agents.

The first use of network explainability for MARL is **Q value path decomposition (QPD)** [10] which uses integrated gradients to map the contributions of inputs from the independent agents to the prediction of the global critic. Integrated gradients are a method used to attribute the predictions of neural networks to their input features and to determine how much influence each component of the input features have on the output value [11]. Unlike QMIX which used the global state s for central value decomposition QPD instead uses $\vec{o}_t$ which is the joint observation vector of all agents as a representation of the global state and $\vec{a}_t = \mathbf{u}$ as a vector representing the joint action of all agents.

3 Relevance Decomposition Network (RDN)

We propose RDN to perform credit assignment between ad-hoc agents in the cooperative multi-agent setting under the assumption of a linear relationship between the individual local rewards and the shared global reward. However unlike most central training decentralised execution (CTDE) methods which uses learned decomposition as an end-to-end system RDN separates learning of the global reward function from the local Q values of the agents.

The details of the algorithm are shown in Algorithm 1. Lines 2-9 show how the decentralised agents interact with the environment to gather data where $Q^i(o_{t,i,.})$ is the Q value of independent agent i at timestep t, h_t^i is the hidden state of agent i at timestep t, $o_{t,i,.}$ is the local observation of agent i at timestep t,$a_{t,i}$ is the action taken by agent i at timestep t and $\pi_i(Q^i(o_{t,i,.}), \epsilon(e))$ is the policy π of agent i dictated by a Q value function and an $\epsilon - greedy$ exploration strategy.

Lines 11 to 15 calculate the expected total Q value at each timestep using a critic network parameterised by θ^c and a target Q value using a target critic parameterised by $\tilde{\theta^c}$ whose parameters are copied over from the the critic network every 200 updates. The critic network is updated using loss $L(\theta^c) = E_{\vec{o},\vec{a},r,\vec{o}'}[(Q_{tot}^{\theta^c}(o_1, ..., o_n, a_1, ..., a_n) - y)^2]$ where $y = r + \gamma(Q_{tot}^{\theta'^c}(o_1^{'}, ..., o_n^{'}, a_1^{'}, ..., a_n^{'}))$ and θ^c is the critic's parameters and $\theta^{'c}$ is the target critic parameters, which are reset every C training epochs

Lines 16 to 21 update the independent agents' parameters θ^i. The total expected Q value for each timestep is decomposed into independent target Q values $\tilde{\theta^i}$ and the DRQNs [12] which act as the agent networks are trained using the loss $L(\theta^i) = E_{\vec{o},\vec{a},r,\vec{o}'}[(Q^{i,\theta^i}(o_i, a_i) - \tilde{Q}^i)^2]$ [13] and $\tilde{Q}^i = \sum_i R_{in}$ where $\tilde{Q}^i$ is the Q target for agent i and $\sum_i R_{in}$ is the sum of all relevance values associated with agent i.

A characteristic of LRP is it maintains a conservative calculation between layers of the neural network [5]. A such the relevance values from later layers are included completely in the calculation of the relevance values of the layers closer to the input. Essentially we can assume that the total sum of the relevance vaues in approximately the same as the output of the NN when LRP is used. therefore we can equate Q_{tot} to the sum of relevance scores as $Q_{tot} \approx R_{in}$.

Algorithm 1: Relevance Decomposition Network (RDN)

```
1  Initialize Critic network $\theta^c$. target critic $\tilde{\theta^c}$, DRQN $\theta^\pi = (\theta^1, ..., \theta^n)$
2  for each training episode $e$ do
3      $s_0$ = initial state, $t$ = 0, $h_0^i = 0$ for each agent $i$
4      while $s_t \neq$ terminal and $t < T$ do
5          $t = t + 1$ for each agent $i$ do
6              $Q^i(o_{t,i,.}), h_t^i = DRQN(o_{t,i}, h_{t-1}^i; \theta^i)$
7              Sample $a_{t,i}$ from $\pi_i(Q^i(o_{t,i,.}), \epsilon(e))$
8          Execute the joint actions $(a_{t,1}, a_{t,2}, ..., a_{t,n})$
9          Receive the global reward $r_t$ and next state $s_{t+1}$
10     Add new episode to replay buffer and sample a batch of episodes
11     for $b$ in batch do
12         for $t = 1$ to $T$ do
13             Calculate targets $y_t$ using $\tilde{\theta^c}$
14             Calculate expected values $Q_t^{tot}$ using $\theta^c$
15     Update critic parameters $\theta^c$ with loss $L(\theta^c)$
16     for $b$ in batch do
17         for $t = 1$ to $T$ do
18             Calculate independant Q values $Q^{i,\theta^i}$
19             Calculate total Q target from updated critic parameters $\theta^c$
20             Using Layerwise Relevance Propagation decompose the total Q target into
                 independent Q targets $\tilde{Q}^i$
21     Update DRQN parameters $\theta^\pi$ with loss $L(\theta^\pi)$
```

4 Experiment Methodology

We use two environments to test our algorithm: a simple 2-step matrix game and the Starcraft Multi Agent Challenge (SMAC). We use the matrix game to show how even in trivial environments methods like QMIX [9] and VDN [8] which rely on a learned decomposition can learn an inaccurate relationship between global and local variables. SMAC allows the creation of more complex cooperative environments. We use a preexisting *bane_vs_bane* which QMIX and VDN have shown difficulty in achieving consistently high performance [2]. We use this environment to show how increasing amounts of redundant agents decrease performance of value deposition algorithms. We also design three additional environments with different numbers of redundant agents.

4.1 Baselines

We consider several algorithms for use as baselines in our experiments. Independent Q learning (IQL) represents a naive approach solving multi-agent tasks but is still able to show reasonable performance in certain environments [7]. QMIX is currently the most popular algorithm to our knowledge and many newer MARL algorithms like ROMA [14] and MAVEN [15] are auxiliary augmentations to build onto it for more specialised use cases. For our experiments we also test against a version of QMIX using the same observation space as our agent to show how the concatenation of local observations is not an effective substitute for the ground truth state. We refer to this as as QMIX_NS.

4.2 Training environments

2-step matrix game

To show the value of using layerwise relevance propagation LRP as a method of direct decomposition rather than learning decomposition we use a simple 2-step cooperative matrix game with only 2 players.

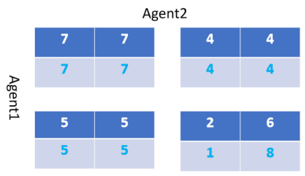


Figure 1: 2 player matrix game: player 1 chooses a row and player 2 chooses a column per step for 2 time-steps. The location of the players in the final timestep determines the global reward assigned to the episode

In step 1 agent 1 chooses a row and agent 2 a column simultaneously. These choices determine the matrix game to be played. In step 2, agents again choose a row and column within the chosen matrix game. The agents receive the payoff of the chosen index as a shared global reward as shown in figure 1.

Independent Q learning methods struggle to solve these simple tasks as agents do not have communication methods to determine the actions of each other [9]. As a result they should favour choosing game 1 which has the highest average payoff irrespective of the action of the other agent rather than game 3 which presents a higher potential reward but relies on coordinated decision making.

Starcraft multi agent challenge (SMAC)

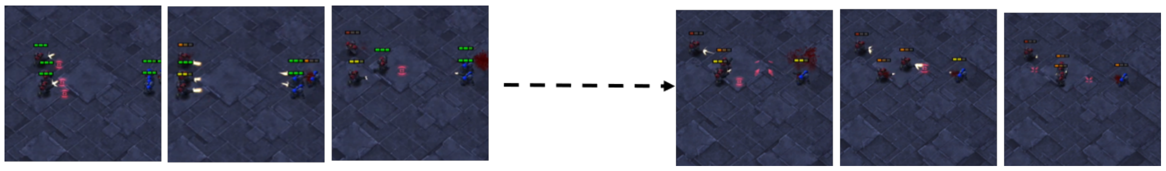


Figure 2: Example SC2 gameplay

SMAC uses Starcraft 2 2 in order to create a series of micromanagement tasks using decentralised controllers [2]. Enemies are controlled by predesigned heuristics. Proper micromanagement is required to maximise the damage dealt to enemy units and achieve the optimal reward. Units are required to learn a wide array of skills including focusing fire, kiting and efficient pathing. It has become a common benchmark for many state-of-the-art (SOTA) MARL algorithms like COMA, QMIX and QTRAN [16][9][17].

4.3 Network architecture and hyperparameter configurations

For the SMAC environments and the 2-step matrix game we use the same hyper parameters used in the original SMAC benchmarking paper [2]. The architecture of the agents is a standard deep recurrent Q network (DRQN) [12] with a gated-recurrent unit (GRU) layer with a 64 cell hidden state followed by 2 fully connected layers with 32 nodes each and a final fully connected layer with nodes equal to the size of the action space of an individual agent.

The critic network is a feed forward neural network with 2 dense layers of 64 nodes each and an output layer of size 1.

We use a γ value of 0.99 in all environments. An $\epsilon - greedy$ exploration strategy is used with an initial ϵ value of 1 which is annealed to 0.05 over 50k timesteps for SMAC and 2000 steps for the matrix game.

To increase learning speed we share parameters across all individual Q networks and use a single network to represent all agents. A one-hot encoding of the agent type is concatenated into each agent's observations. By using the agent type as part of the observation space we can train a single shared neural network to represent all agents which has been shown to improve learning speed significantly with minor performance loss [18]. Both the critic and the agent networks are trained using Adam with a learning rate of $5\text{x}10^{-4}$. The replay buffer for SMAC contains the most recent 1000 trajectories and is sampled with a batch size of 32. For the 2-step matrix game we only keep 200 trajectories and use a batch size of 4. Target networks for the critic are updated every 200 training epochs.

We test our method every 100 training episodes for SMAC and 10 for the matrix game for 20 test episodes with exploration disabled. We evaluate performance based off of the percentage of wins per set of test episodes. We conduct 8 separate training runs in order to gather the date from the experiments.

Agent networks are given an input consisting of only the most recent observation, previous action and their identity as a one-hot encoding. The critic for RDN is given a concatenation of all agents' previous 4 observations, current actions and their one hot encoded identities. For QMIX the critic takes in the ground truth state and the expected Q values of the independent agents as in the original QMIX paper [9].

5 Results

2-step matrix game

All methods are quick to come to convergence on this simple environment. However our method is the only one able to fully converge on the correct solution with no variance as seen in the figure 3. The independent Q learning method overall converges on the better policy than QMIX as training progresses. This could indicate that the independent method is able to more effectively perform exploration due to training of agents not being linked to the central critic. As QMIX agents are reliant on the factorised signal from the central critic if the critic is performing inaccurate decomposition then agents are unable to learn an effective policy as they are receiving a poor representation of the actual reward signal. For independent agents the reward signal is based on the join policy at each timestep therefore even without value factorisation agents are able to eventually determine a higher reward policy.

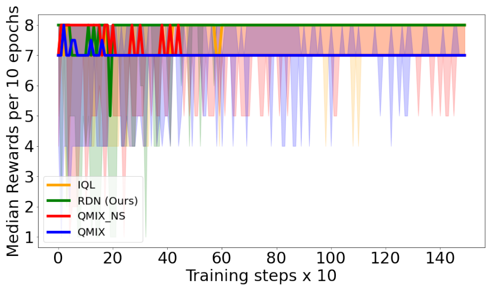


Figure 3: Median reward from 8 training runs on a 2 step grid world where shaded region is the 25th to 75th percentile range of rewards

Starcraft2

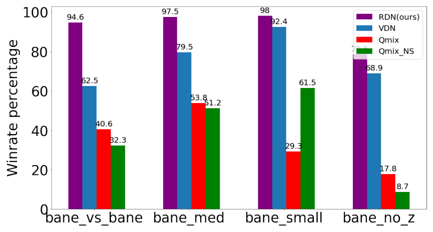
We evaluate our algorithm in scenarios where all agents in the team are of the same unit type (homogenous) and where different agents of different unit types (heterogenous) must collaborate to solve the scenario.

The preexisting map from *the Starcraft Multi-Agent Challenge* (SMAC) we make use of is the *bane_vs_bane* map. In this map each side has 20 zerglings and 4 banelings. The most optimal policy for this environment has the zerglings move out of the way so as to not obstruct the banelings' movement. We make us of 3 additional variants of the original map. *bane_med* which only has 15 zerglings, *bane_small* with has 10 zerglings and *bane_no_z* with has no zerglings. We use a diminishing number of redundant agents in order to show how unneeded agents attempting to form cooperative policies is actually detrimental to the overall policy of the environment. Only *bane_vs_bane* is a symmetrical environment meaning the same number of each type of enemy and ally agent are present. In all other cases we use the original parameter settings from *bane_vs _bane* for the number of enemy units spawned while only varying the allied unit composition.

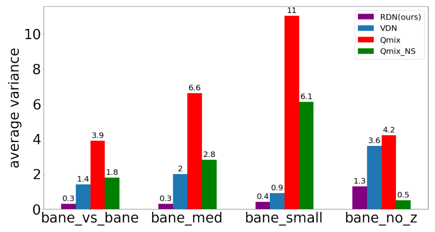
From figure 4a we can see that across all maps RDN outperforms all baselines although the performance of VDN improves as the number of redundant agents are reduced. Interestingly QMIX is outperformed by QMIX_NS in *bane_med* and in *bane _no_z* indicating that in cases where there are an intermediate number of redundant agents the central state already begins to become uninformative making accurate multi-agent credit assignment difficult.

From figure 4b QMIX is shown to have significantly higher average variance than all other algorithms which only use the local inputs. This is likely due to the increasing size of the ground truth state with respect to the increasing number of agents required to cooperate under the environments. QMIX_NS obtains low variance in *bane _no_z* however this is due to achieving poor results overall and therefore low fluctuations in the rewards achieved. Figures 5a and 5b confirm this as we can see for RDN and VDN the difference between the 75th and 25th percentile winrates is minor as both algorithms are unaffected by an increasingly complex state space as the number of redundant agents increases. Comparatively QMIX displays large variance on all 4 environments as the high number of redundant agents makes the central state difficult to interpret.

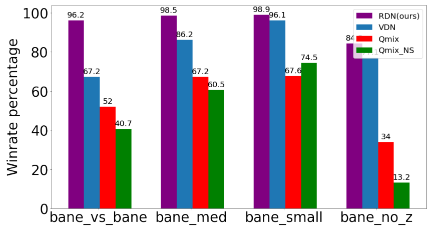
In the case where all redundant agents are removed VDN performs to a similar level to RDN.



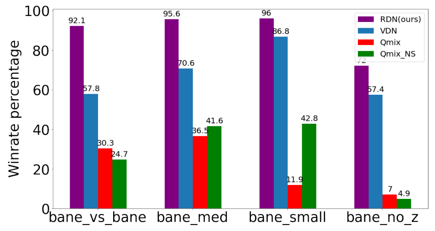
(a) Percentage winrates from highest number of redundant agents (left) to least redundant agents (right) for all tested algorithms



(b) Variance from highest number of redundant agents (left) to least redundant agents (right) for all tested algorithms



(a) 75th percentile winrates from highest number of redundant agents (left) to least redundant agents (right) for all tested algorithms



(b) 25th percentile winrates from highest number of redundant agents (left) to least redundant agents (right) for all tested algorithms

6 Conclusion

In this paper we show how increasing numbers of redundant agents makes reaching stable convergence to an optimal joint policy difficult for both monotonic factorisation methods like QMIX and linear factorisation methods like VDN where only n of m total agents are required for an environment to be solved. Also we show how these methods converge to suboptimal minima even in simple matrix games.

We propose RDN as a method that uses LRP in order to perform more optimal credit assignment in environments with high numbers of redundant agents using only local agent observation. RDN is able to reach near optimal convergence on all environments used with similar overall winrates without the use of the ground truth state information. With VDN and QMIX we see a gradual decay in performance as the number of redundant agents increases.

An ablation was done comparing the use of a fully observable observation space to one with only the standard space. In the case of explaining relevance directly from inputs fully observable information is unneeded as it has no influence on the decision making of the local agents. This is further verified as QMIX is unable to solve the simple matrix game and raises the question of why the central state is needed for QMIX to operate effectively and if the ground truth state is entirely needed to create high performing deep MARL agents [19].

References

[1] J. T. Schwartz and M. Sharir, "On the "piano movers'" problem i. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Communications on pure and applied mathematics*, vol. 36, no. 3, pp. 345–398, 1983.

[2] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, "The StarCraft Multi-Agent Challenge," *CoRR*, vol. abs/1902.04043, 2019.

[3] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

[4] Y. Yang, J. Hao, B. Liao, K. Shao, G. Chen, W. Liu, and H. Tang, "Qatten: A general framework for cooperative multiagent reinforcement learning," *arXiv preprint arXiv:2002.03939*, 2020.

[5] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-wise relevance propagation: An overview," *Explainable AI: interpreting, explaining and visualizing deep learning*, pp. 193–209, 2019.

[6] C. Amato, "Decision-making under uncertainty in multi-agent and multi-robot systems: Planning and learning," Jul. 2018, pp. 5662–5666. DOI: `10.24963/ijcai.2018/805`.

[7] J. N. Foerster, "Deep multi-agent reinforcement learning," Ph.D. dissertation, University of Oxford, 2018.

[8] 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 *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18, Stockholm, Sweden: International Foundation for Autonomous Agents and Multiagent Systems, 2018, 2085–2087.

[9] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *ICML*, 2018.

[10] Y. Yang, J. Hao, G. Chen, H. Tang, Y. Chen, Y. Hu, C. Fan, and Z. Wei, "Q-value path decomposition for deep multiagent reinforcement learning," *arXiv preprint arXiv:2002.03950*, 2020.

[11] M. Sundararajan, A. Taly, and Q. Yan, *Axiomatic attribution for deep networks*, 2017. arXiv: `1703.01365 [cs.LG]`.

[12] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *arXiv preprint arXiv:1507.06527*, 2015.

[13] T. W. Sandholm and R. H. Crites, "On multiagent q-learning in a semi-competitive domain," in *International Joint Conference on Artificial Intelligence*, Springer, 1995, pp. 191–205.

[14] T. Wang, H. Dong, V. Lesser, and C. Zhang, "Roma: Multi-agent reinforcement learning with emergent roles," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[15] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "Maven: Multi-agent variational exploration," *arXiv preprint arXiv:1910.07483*, 2019.

[16] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[17] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2019, pp. 5887–5896.

[18] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Comparative evaluation of multi-agent deep reinforcement learning algorithms," *CoRR*, vol. abs/2006.07869, 2020. arXiv: `2006.07869`. [Online]. Available: `https://arxiv.org/abs/2006.07869`.

[19] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. M. Bayen, and Y. Wu, "The surprising effectiveness of MAPPO in cooperative, multi-agent games," *CoRR*, vol. abs/2103.01955, 2021. arXiv: `2103.01955`. [Online]. Available: `https://arxiv.org/abs/2103.01955`.

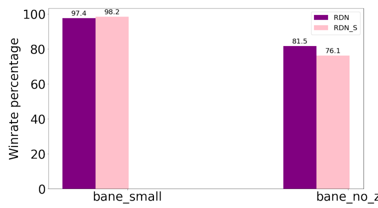
A Appendix

A.1 Effects of observation space

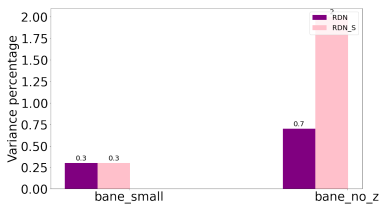
We perform an ablation showing how relevance decomposition network (RDN) is affected by the the availability of state space information. As most central training decentralised execution (CTDE) methods use the ground truth state to condition the value function learnt by the critic we show that state information is not always guaranteed to produce greater performance. For this test we give the critic access to a fully observable input space which includes data that would normally be unavailable during training.

In our testing we find minimal utility from using the expanded state observation space. In 3 of 4 maps show in figure 6a Where performance remains similar across environments for both versions of the RDN.This is likely due to regions of the expanded space being given weight from relevance calculations but not actually having any effect on the decision making of the agent as the agent only uses its own observations during runtime. The revealed areas have no effect on decision making and as such should not be considered relevant.

We also find an increase in the variance of the algorithm when a fully observable space is used as see in figure 6b. As there is a large amount of data that does not effect the runtime policy of the agent used in the learning process decomposition of the value function is being done which assigns relevance to information that does not actually have any correlation with the policy during runtime which reduces the stability of the learning process.



(a) Percentage winrates from most number of redundant agents (left) to least redundant agents (right) for all RDN and RDN_s



(b) Variance from most number of redundant agents (left) to least redundant agents (right) for all RDN and RDN_s