Diverse partner creation with partner prediction for robust K-Level Reasoning

Jarrod J. Shipton
Department of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg
Jarrod.Shipton@wits.ac.za

Benjamin Rosman
Department of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg
Benjamin.Rosman1@wits.ac.za

Abstract

In Multi-agent Reinforcement Learning (MARL) there has been a substantial move towards creating algorithms which can be trained to work cooperatively with partners. In general this is done in a self play (SP) setting, where the agents are set to play and train with copies of themselves in a Decentralized Partially Observable Markov Decision Process setting. Agents trained with SP often result in behaviour such that arbitrary conventions, or ”handshakes”, will be formed in order to more efficiently achieve their goal. These arbitrary handshakes can be seen as unwanted behaviours as they creates the issue that when agents are paired with novel agents they will often not be able to complete a task cooperatively, even when paired with different training runs of the same algorithm. A valuable architecture to help tackle this problem is synchronous K-level reasoning with a best response (SyKLRBR), which creates agents that have policies based on grounded information which are robust to various handshakes. Weaknesses are still shown in that certain agents with specific handshakes still outperform this agent when paired with one another as compared with the SyKLRBR agent. This work expands on the SyKLRBR framework by factorizing the action-observation histories to fit a belief over a diverse set of agents created with multiple different runs of a modified SyKLRBR algorithm. These modifications allow the algorithm to create and identify a robust set of agents with various handshakes that could exist in potential novel partners, ultimately allowing it to take advantage of these handshakes for better results.

Keywords: Multi-Agent Reinforcement Learning
Partial Observability
Zero-shot Coordination
K-Level Reasoning

Acknowledgements

The authors would like to thank the members of RAIL Lab based at The University of the Witwatersrand, Johannesburg for valuable discussion and support.

1 Introduction

In Multi-agent Reinforcement Learning (MARL) there has been a substantial move towards creating algorithms which can be trained to work cooperatively with partners. This is generally done in self play (SP), where the agents are set to play and train with copies of themselves in a decentralized Partially Observable Markov Decision Process (Dec-POMDP) setting.

Agents trained with SP often result in behaviour such that arbitrary conventions over actions will be formed in order to more efficiently achieve their goal. An example of such a convention can be observed in the Simplified Action Decoder for Deep Multi-Agent Reinforcement Learning (SAD) [4] where the agents are trained to play the game Hanabi. Hanabi is a card game comprising of cards with five different colours and five different ranks. Players take turns without communicating, besides through allowed gameplay actions, to play cards from their hands to form five piles of cards with matching colour and increasing rank without making more than three mistakes throughout the game. Each player does not have full information of the contents of their own hands and has to infer what each card is from the actions of other players. Each player may perform one of three main actions on each of their turns: give a hint about cards of specific quality to a single player, play a card, or discard a card. It has been observed that the trained agents used hints for purposes unintended by the game rules, such as, hinting the colour red meant the next player should play their first card instead of just signalling the red cards. Such an arbitrary convention, which we refer to as a ”handshake”, would be a desired and welcomed result if these agents are only to be paired with copies of themselves, or another agent with understanding of these handshakes. However, such a constraint detracts from reality, where the agent could have formed a handshake for any of the available colours, leading to a situation where a novel (previously unseen) partner would not know the handshake, and possibly act on their own, differing handshake leading to an undesired outcome.

It quickly becomes obvious that agents with specific handshakes will perform poorly outside of SP, that is, paired with novel agents. It is not even guaranteed that such an agent will perform well in cross-play (XP), where agents that are trained by the same algorithm, but in different training runs are paired at testing time. Notably, the take away from this is that well-trained SP agents, potentially using their own unique handshakes, will not always perform well with other well-trained, novel partners at test time. In works such as Other Play (OP) [6], Synchronous K-Level Reasoning with a Best Response (SyKLRBR) [3], and Off Belief Learning [5] the authors have developed methods to aid agents to be robust to handshakes and work cooperatively with novel agents at test time, or termed otherwise, aiding them to achieve zero-shot coordination (ZSC). These works show that agents which are robust to handshakes will perform better than those that have been trained with specific handshakes in XP or when paired with novel agents. However, this avoids a factor that can be taken advantage of: some of these novel partners perform better with their own handshake than with robust agents. It should thus be a goal to have an agent that is able to learn a policy that is both robust to many different handshakes, but also capable of identifying any known handshakes and responding to these in an optimal fashion.

In this work we take the idea that agents should be trained with a variety of agents which differ in behaviour to keep them robust to specific handshakes, however, we also wish for the agent to play in a specific manner when paired with an agent that has a similar behaviour to an agent it has trained with previously. To do this we propose a method to create a diverse set of training agents, and we train our agent to identify which of these agents it is playing with so that it can respond to their behaviour it recognizes to be a known handshake with the correct response, while still being robust to various play patterns.

2 Background

2.1 Dec-POMDPs

In the Dec-POMDPs setting we have N agents $i \in \{1, \ldots, N\}$. These agents receive observations $o_t^i = O(s_t)$ which are derived from the underlying state $s_t \in S$ for each agent at each timestep t. The agent then, at each timestep, performs an action, u_t^i from its policy $\pi_i \sim \pi(u^i|\tau_t^i)$, where τ_t^i is the action-observation history (AOH) of the agent. We define $\tau_t^i = \{o_0^i, u_0^i, r_1, \ldots, r_{t-1}, o_t^i, u_t^i\}$, where r_t, in this work, is the shared reward of the environment at each timestep defined by $R(s,t)$, which is the environment's reward function. The goal of the agents is to maximise the total expected reward, $\mathbb{E}_{\tau \sim P(\tau|s,u)}[R_t(\tau)]$. In this context $R_t(\tau)$ is the discounted sum of rewards for the entire AOH, $R_t(\tau) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}$, where γ is the discount factor. In general this could be for an infinite horizon, but in this work the environment is bounded by a maximum timestep limit of t_{max}.

2.2 SyKLRBR

The method SyKLRBR is a modified combination of two previously studied ideas, Cognitive Hierarchies (CH) [1] and K-Level Reasoning (KLR) [2], which deal with how to create a policy that will produce the best response (BR) to another policy's actions. The BR is the action taken by an agent which leads to the maximal reward given the other agent's action.

In CH k agents are trained to act in a given environment. The first of these agents is trained to act in the environment with versions of itself and is referred to as the first of k-level agents. Each of the subsequent k-level agents are trained with versions of themselves and a distribution of the previous $k-1$-level agents. A Poisson distribution of the previous k-level agents is commonly used. Each will then have a policy, π_k, which is a best response to the all previous k-level agents. In KLR agents are trained in a similar fashion, however they are only trained as a BR for the k-level policy directly before it, $k-1$-level. In both of these methods the agents of each k-level are trained until a predefined halting point, and only then is the next level trained.

A Recurrent Replay Distributed Deep Q-Network (R2D2) [7] is used in SyKLRBR to train all k policies, π_k, synchronously, initializing the parameters, θ, of all policies randomly. The first policy of the k-level policies, π_0, picks an action randomly from each of the available actions each turn, with each action having the same probability. That is, the action is chosen at random from a uniform distribution over the action space. This ensures that there is no implied handshake in any of its actions, meaning all information it shares through its actions is grounded information. The policies for $k \geq 1$ are then trained with this as their $k=0$ policy, sampling training partners from the k-levels below it using a Poisson distribution as in CH. The synchronous training is shown to produce a more robust agent than when training the levels sequentially as in KLR.

3 Proposed Method

Our method proposes that the policy of the agent should depend on its belief of the agent type it is playing with as well as on the AOH. In order to have a belief over the types of agents we are playing with, we would need to have a set of diverse agents to train with and for the environment episodes to be sufficiently long such that behaviour can be observed to form such a belief. We thus need to produce agents which are diverse, and keep a belief over which of these agents our agent is playing with.

In a Dec-POMDP setting the policy of an agent is dependent on its AOH, τ_t^i. Our work proposes that this should instead be based not only on the AOH, but with the belief of the agent type it is playing with as well. In order to infer a belief of the type of agent we are working with, we need to fit this belief only to observations and actions the other agent executes. Thus the belief is based on a distribution dependent on a factorized view of the AOH τ_t^i. The specific factorization of the AOH is into a public AOH, $\tau_t^{i,pub}$, and private AOH, $\tau_t^{i,priv}$. The public AOH contains all information about the trajectory which is available to be observed by all agents at each time step t, while the private AOH is the information only observable to agent i at each time step t. Using τ_t^{pub} we fit a distribution, $\Psi(\psi_t|\tau_t^{pub})$, where ψ_t is the belief of which of the agents we are playing playing with. This distribution is learned using an LSTM and is trained alongside the main policy, $\pi_i \sim \pi(u^i|\tau_t^i, \psi_t^i)$ which now incorporates ψ_t^i. A visualisation of this policy can be seen in Figure 1a.

To produce the set of diverse agents with which ours trains, we train M different SyKLRBR training cycles replacing their policy structure with ours. That is, we train $M \times K$ different policies $\pi_k^m \sim \pi(u^k|\tau_t^k, \psi_t^k)$. In each training run $m \in M$ the policy π_0^m is set to have randomized and differing probability weights for each action in the action space resulting in M differing sets of k-level agents. This is a particularly useful feature of the SyKLRBR training structure since for a given policy π_0^m the kth level policy will differ from the policy of another training run m, provided the initial distribution for π_0^m between the two runs m differs enough. The final policy, π_{k+1} is then trained uniformly with each of the sets of M agents, each set having a Poisson distribution over its respective k-levels. This has been visually represented in Figure 1b.

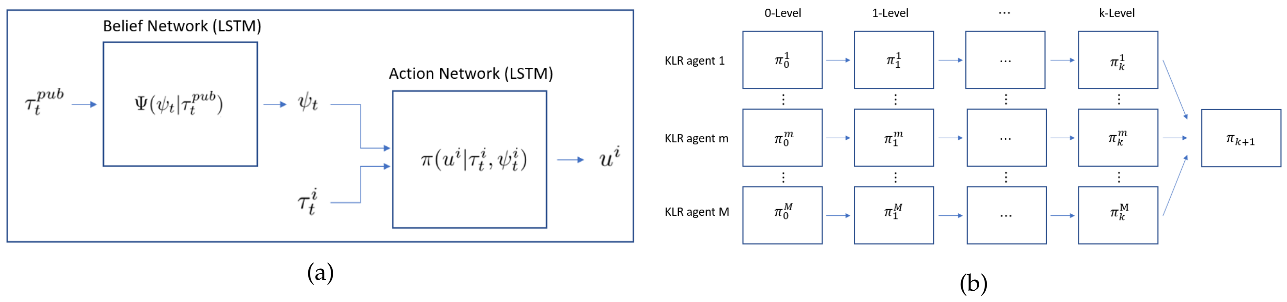


Figure 1: a) The structure of the policy network. Using the public AOH as the input for the belief network, we produce the belief ψ_t, which is then paired with the AOH as inputs to the action network which chooses an action u^i. b) The structure of the M K-level policies with the K+1-level final policy. Each row represents a SyKLRBR policy run, with each arrow showing a preceding level of that run. Each successive policy is trained with a Poisson distribution of the previously connected policies, and π_{k+1} being uniformly paired with each of the m training runs.

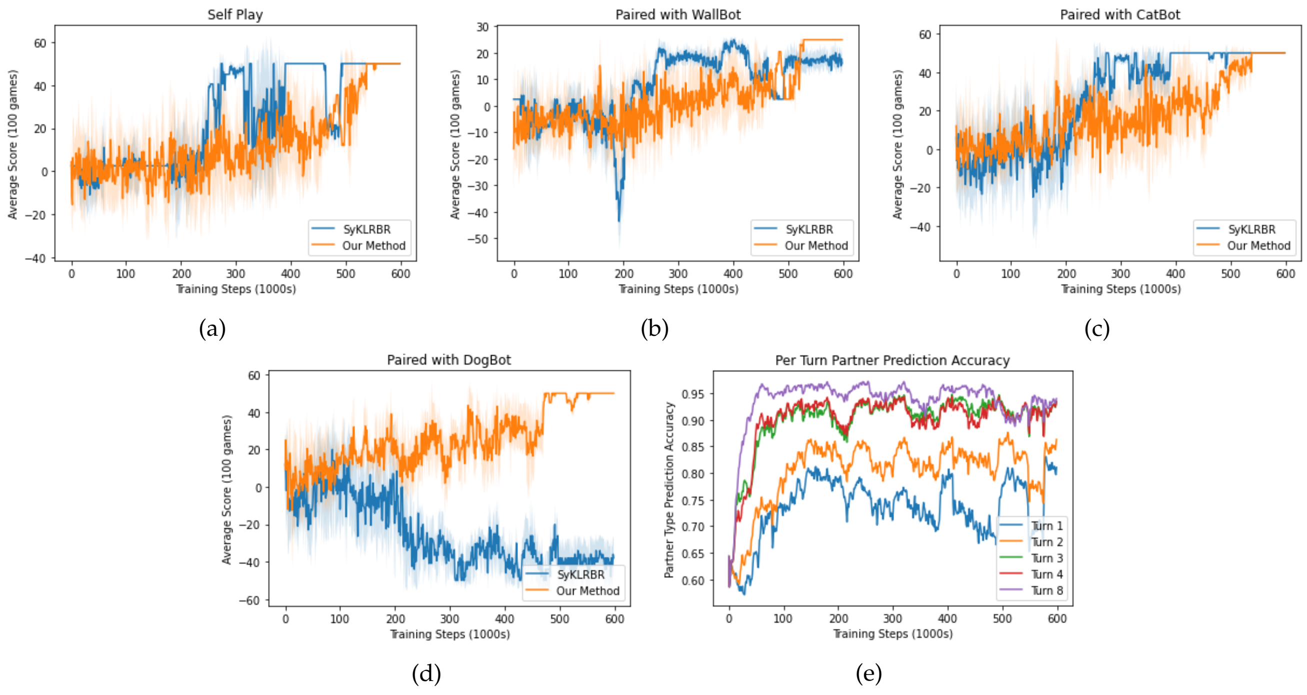
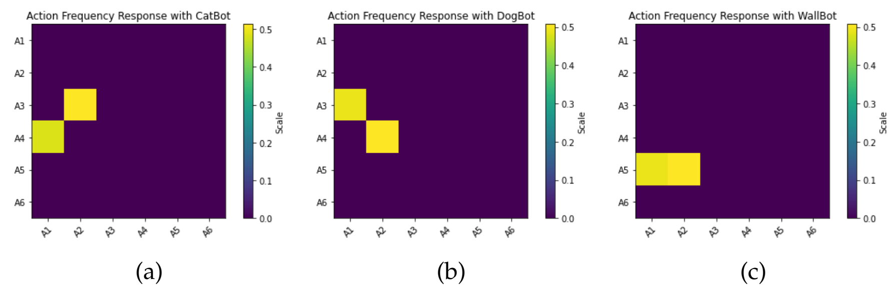


Figure 2: The comparison in scores between our proposed method and SyKLRBR during training is shown for (a) self play, (b) paired with WallBot, (c) paired with CatBot, and (d) paired with DogBot. (e) The per turn partner prediction accuracy during training of our proposed method.



A1	Guess Dog
A2	Guess Cat
A3	Turn Light on
A4	Turn Light off
A5	Lower wall
A6	Bail out of game

(d)

Figure 3: (a) The action response pairs of our proposed method when paired with CatBot. (b) The action response pairs of our proposed method when paired with DogBot. (c) The action response pairs of our proposed method when paired with WallBot. (d) Action reference table.

4 Evaluation

To evaluate this architecture and to allow for a prolonged interaction to create a belief of the player type we take a repeated version of the toy environment from OBL [5], repeating the game 5 times for each episode. This is a team game which involves two players with shared reward. In this game, there is a binary variable $pet \in \{dog, cat\}$ which player 1 can observe, however player 2 cannot due to a wall between the two players. This wall can be lowered by player 1 to reveal to player 2 which pet is with player 1. If this action is taken all reward values are halved, resulting in a maximum reward of 5. Furthermore there is a light which can be turned on by player 1 and observed by player 2, which is provided as a communication avenue to allow for the formation of handshakes beyond that of just lowering the wall. The goal of the game is for player 2 to correctly guess what pet player 1 has: if they guess correctly the team of two get a reward of 10, if they guess incorrectly the team gets a reward is -10. There is also a third option of avoiding the guess and to bail out of the game for a team reward of 1.

For the purpose of testing our algorithm on this environment we created three rules-based bots: one which always lowers the wall, and chooses the correct option if the wall was lowered (WallBot), one which turns on the light if the pet is a cat, and picks cat if the light is turned on (CatBot), and the third turns on the light if the pet was a dog, and picks dog if the light is turned on (DogBot). Our training run using our proposed method had $M = 4$ and $K = 4$. We trained a SyKLRBR version of the agent, with $K = 4$ for comparison. We found that both our algorithm and SyKLRBR achieved 50 points in

SP. However, when we tested these trained agents with the rules based bot, a difference in results was apparent, as seen in Figures 2a-d. The standard SyKLRBR agent achieves a perfect score of 50 points with either CatBot or DogBot, and a negative score with the other, occurring with an approximately even distribution across training runs. The SyKLRBR agent trained for this comparison, scores 50 points with CatBot, and -38 ± 5.9178 with DogBot. Our proposed method consistently achieves 50 points with both CatBot and DogBot.

Furthermore it can be seen that when our bot accurately predicts the agent it is playing with, the action it responds with matches the convention of that agent, as is shown in Figure 3. In Figure 3a we observe that for the CatBot, the response to turning on the light (action A3) is guessing cat (action A2), and the response to seeing a cat is turning on the light (action A3) with the CatBot responding by guessing cat (action A2). Similarly the correct responses can be observed for DogBot and WallBot.

While identifying the agent type is important, there is no accurate measure of how well it can predict if it is playing with CatBot, DogBot or WallBot. This is due to the fact that it is trained to identify which of the $M \times K$ partners it is playing with, and the accuracy of the prediction can lower if any of those $M \times K$ partners behave similarly at the end of training, which is probable for an environment such as this one where there are three optimal strategies. However, if the behaviour is similar between training partners, the best response will be similar, and thus while the measured accuracy decreases, it does not necessarily adversely affect the final agent. The accuracy of the belief of which partner the final agent is playing with increases over the course of the game and is shown, on a per turn basis, in Figure 2e.

In this environment we have discovered this algorithm is capable of learning to identify diverse agents created by the M training runs of our proposed architecture, identifying their type after as little as 4 actions, which is only possible due to the repeating version of this toy environment, which would otherwise end in 2 actions. It is important to reiterate that, in line with the assumptions of the model, this model requires an environment in which trajectories of a minimum sufficient length exist, to identify behaviour of these agents, acting in a similar fashion to how an agent trained by regular SyKLRBR would until it has successfully identified the agent type it is partnered with, after which it behaves in a manner that is able to play successfully with that agent.

5 Conclusion

By using M different distributions over the action space to initialize π_0 for each of the M SyKLRBR training runs, we are capable of creating a diverse set of agents. The addition of the belief network allows the agent to fit AOH trajectories to specific agent types. With these features our proposed method becomes both the source and identifier of diverse agents. These agents can then be used as training partners during the training process of this algorithm resulting in a robust agent that is capable of identifying the type of partner it is working with. This allows it to be able to match the handshake of an unknown agent if its behaviour mimics that of a known agent, while still maintaining a high level of robust behaviour for agent types it has not come across. This does come with the cost of long initial training times. These preliminary results on the toy environment presented in this paper are promising and offer an exciting avenue to apply this to environments such as Hanabi.

References

[1] Colin F Camerer, Teck-Hua Ho, and Juin-Kuan Chong. A cognitive hierarchy model of games. *The Quarterly Journal of Economics*, 119(3):861–898, 2004.

[2] Miguel A Costa-Gomes and Vincent P Crawford. Cognition and behavior in two-person guessing games: An experimental study. *American economic review*, 96(5):1737–1768, 2006.

[3] Brandon Cui, Hengyuan Hu, Luis Pineda, and Jakob Foerster. K-level reasoning for zero-shot coordination in hanabi. *Advances in Neural Information Processing Systems*, 34, 2021.

[4] Hengyuan Hu and Jakob N Foerster. Simplified action decoder for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1912.02288*, 2019.

[5] Hengyuan Hu, Adam Lerer, Brandon Cui, Luis Pineda, Noam Brown, and Jakob Foerster. Off-belief learning. In *International Conference on Machine Learning*, pages 4369–4379. PMLR, 2021.

[6] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

[7] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2d2: repeatable and reliable detector and descriptor. *arXiv preprint arXiv:1906.06195*, 2019.