Training human-AI agent in Overcooked

Jim Vos
Supervisors: Robert Loftin, Frans Oliehoek
EEMCS, Delft University of Technology, The Netherlands

June 19, 2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

Most cooperative games are tackled by creating a team of agents who are optimised for each other and the problem. Creating an agent who can play in a variety of teams without any foreknowledge of its partner is a different challenge. These AI systems could useful for human-AI interaction as different people bring a lot of variance into the system. The training method synchronous K-level reasoning best response (SyKLRBR) tries to tackle this problem by creating agents based on grounded information. This research tested the potential of SyKLRBR in human-AI research evaluating the performance of its agent in the cooperative game Overcooked. The agents were able to obtain consistent scores against several unseen strategies, suggesting that SyKLRBR is able to create robust agents. Where SyKLRBR shows potential for medium cooperative settings this paper also discusses its great weakness for highly cooperative problems.

1 Introduction

As artificial intelligence (AI) is becoming a bigger part of our lives people have to learn how to deal with these new systems. The AI systems equally so have to learn how to interact with humans. State of the art AI's are good at optimising performance for a singular task but struggle to generalise over new environments, which becomes a problem when a single AI has to deal with the variance of human behaviour. Cooperative environments seem more demanding for human-AI interaction compared to the successful AI's in competitive environments like the games Go and chess [1]. Research into cooperative AI could help to improve the increasing amount of interdependent human-AI relationships. The goal of this research is to find out how current training methods can be improved to create more robust agents.

This paper will describe further research on the training method SyKLRBR [2] in cooperative play. Cooperative play is often solved as a multi-agent learning problem (MARL) creating a team of agents that are optimised for each other and the task. These teams often converge to strategies only they can understand by generating arbitrary conventions, "handshakes", making them perform worse outside of self-play (SP). In this research, the focus is on ad-hoc teamwork [1] where a single agent must be capable of cooperating with different unseen partners. SyKLRBR tries to create such agents by training a population in a hierarchical structure. Each level in the hierarchy is trained simultaneously creating new agents in every training session. By constantly training with new agents the model should perform better in ad-hoc play.

The paper [2] does show the effectiveness of SyKLRBR-trained agents in ad-hoc play in the cooperative game Hanabi. The results show that these agents can generalise a cooperative strategy with unseen agents. This could be promising for modelling agents that need to cooperate with different humans. Although SyKLRBR [2] is promising, it is also limited as the training algorithm has only been tested on the game Hanabi. This paper will test SyKLRBR on the Overcooked environment [3]. This environment is a simplified version of the cooperative game Overcooked optimised for human-AI research. In order to see if SyKLRBR has its flaws or is actually a promising training method, this paper will try to answer the following question.

- **"Does an agent trained with SyKLRBR perform well in ad-hoc play in Overcooked?"**

2 Related work

2.1 MADDPG

Multi-agent learning problems are a field of AI that has not seem as much exploration compared to competitive coordination settings. Current training algorithms such as policy gradient or Q-learning are not suited for MARL. Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [4] is a version of policy gradient where the agents learn centralised, uncovering general strategies. During test time the agents only have access to their local observation space making them suited for ad-hoc play. The goal of the agents is to uncover the strategies of their partners at test time, so they are able to coordinate without communication. The amount of strategies MADDPG can uncover is heavily dependent on the training time making it expensive to create a general strategy.

2.2 Other-play

The game Hanabi has been used for other researches on ad-hoc play. In this paper [5] a training method "Other play" shows improved performance in multiple zero-sum games including Hanabi. Other-play tries to break symmetries of the self-play agent. Symmetries are conventions the agent expects and thus could react upon. The training algorithm is based upon self-play where in other-play the agent trained against has randomly altered states and actions for known symmetries. Since the agent will not have the power to change both of their models predicatively it has to focus on its own. This will lead to an agent that is robust to different and unpredictable agents leading to better ad-hoc performance. For this algorithm to work you need to know the symmetries in advance. In overcooked the environment changes and so do the optimal decisions, which makes it hard to find symmetries. Other-play is thus not perfectly suited for ad-hoc play in Overcooked.

2.3 Off-belief

Off-belief [6] is an algorithm developed to generate agents for ad-hoc play. It tries to understand the game by looking at past decisions in order to generate a new policy at every future action. The policy-making is only based upon grounded information meaning it will not make arbitrary conventions which would fall off in ad-hoc play. Off-belief is not a silver bullet for zero-shot coordination as it works best for turn-based environments[6]. Simultaneous-action games can be converted to turn-based games by limiting the observer's view from the current round. Since time was limited during this research changing Overcooked was out of scope.

2.4 Overcooked environment

The overcooked environment [1] is already used to test human-AI interaction. Originally the overcooked environment [3] was set up to create agents for human-AI interaction with the help of human input data [1]. A behavioural cloned agent is used to train an agent via reinforced learning, creating an agent that has improved performance with humans. The creation of human-like agents is not optimised and not possible for every environment, thus the method is not perfect. Other approaches to human-AI coordination in Overcooked

often use population-based trained agents, which are able to generalise a strategy without unique handshakes. One of the methods is fictitious co-play [7], population-based training is performed on self-play trained agents to generalise their strategies. Fictitious co-play seems to generate agents that perform better in ad-hoc play compared to self-play or population-based trained agents apart.

3 Background

3.1 Zero-Shot Coordination

Different evaluation methods can give varying results on the performance of an agent. For testing ad-hoc play the performance against novel agents is most relevant. Zero-shot coordination (ZSC) is a formalisation of ad-hoc play where the coordination of independently trained agents is evaluated at test time [5]. All agents are familiar with the (partially) visible environment in which they all generate a strategy over which they cannot coordinate. For an agent to perform well in ZSC, it must be robust and cannot rely on previously learned handshakes. Strategies which are based on grounded information from the environment thrive in the ZSC setting, as the information is consistent over the training and testing phase. For human-AI coordination we are looking for generalised strategies, and thus ZSC is very useful as it filters agents models defined on the core principles of coordination.

3.2 K-level reasoning

K-level reasoning is a hierarchical model that originates from the economical world. It was created to formalise human behaviour in the stock market.[8]. K-level reasoning is best explained as an example of "The Keynesian beauty contest". In this contest, participants need to send in a number that is half of the average of all submissions. You could reason that the numbers will be uniformly random as the contestants will have no strategy, giving an average of 50. A second group will reason that all people will send random numbers and thus will send in 25 to get half of the average. A potential third group believes that most people will pick the strategy of group 2, therefore they will send in 13. This process can go on until the best option is 0. The choice is thus dependent on how intelligent you expect your opponents to be. John Kenyes stated that this phenomenon was also occurring in the stock market. The prices of assets are not determined by what investors think it's worth, but by what they believe others think it is worth, which again spirals up the k-levels of reasoning.

This level of reasoning can be applied to humans playing cooperative games like Overcooked. Players will personally fit into the hierarchical scheme by their difference in skill and strategy, and so does the AI. In the games where close cooperation is needed, an understanding of your partner is needed in order to form the 2 strategies into one. As human players infer the AI's skill from a random player to an advanced one, it will pick a level in the k-level hierarchy. The AI does the same where preferably they choose layers directly next to each other.

3.3 SyKLRBR

SyKLRBR is a training algorithm that trains an agent with limited training partners while preventing the creation of conventions between agents. SyKLRBR is an acronym for syn-

chronous k-level reasoning best response, describing the methods used for the algorithm. SyKLRBR algorithm uses k-level reasoning to define the training partners of the agents. The agent on level-k,π_k, trains against all policies below $\{\pi_k, ..., \pi_0\}$. During training the training partner get picked via the Poisson distribution $P(x) = \frac{e^{-\lambda}\lambda^x}{x!}$ where $\lambda = 1$. The model of the lowest level π_0 is static, and its actions are randomly generated from a uniform distribution over the action space. This prevents agents to generate handshakes from predictable behaviour, which is harmful in the ZSC setting. The k-level agent performs better in ZSC settings compared to the levels below [1] as it encounters the most strategies, thus its model is used for creating the final agent.

The second part of SyKLRBR is to synchronously train all the levels in the hierarchy in order to prevent overfitting on fixed strategies. When agents are sequentially trained the partners will have a fixed model. This can lead to predictive behaviour on which the agent will generate handshakes which will be obsolete in ad-hoc play. Synchronous training simulates multiple agents by constantly changing the model during training. The mutation of the agents will ensure that more different states will be encountered during the training phase. This exploration will give more options to detect the grounded information which is useful in the ZSC setting.

The last 2 letters of SyKLRBR stand for best response, this means that the agent deterministically chooses the action it thinks will lead to the highest reward. The algorithm only sees the current state of the environment while picking an action, also known as a feed-forward neural network [9]. As previous steps of the partner are not recorded the agent is not able to make inferences on the partner's strategy during play time. The partner's action only makes up a small part of the observable environment making most information grounded. Creating a best response to the grounded information not only helps to create strategies independent of the opponent, but also makes the agent predictable which can be useful for humans who are able to remember previous actions.

4 Methodology

The main goal of this research is to test the potential of SyKLRBR in the cooperative setting of Overcooked. SyKLRBR did show a performance increase the performance in ad-hoc teamplay compared to Off-belief [6] and Other-play [5]. The Overcooked environment[3] is a fully collaborative game with ZSC test settings used for human-AI coordination research. Testing the performance of SyKLRBR outside the game of Hanabi gives more diverse data on its ability to coordinate in ad-hoc play, but also shows the algorithm's robustness to different environments. Using SyKLRBR trained agents in the Overcooked environment will show the potential usefulness of the algorithm and the usefulness of the method in human-AI coordination problems.

4.1 Experimental setup

The game Overcooked is played with 2 players trying to serve as many meals as possible within a time limit. Every round the 2 players independently choose an action from the six actions $\{move\,up,\ move\ down,\ move\ left,\ move\ right,\ do\ nothing,\ interact\}$ which are simultaneously performed. Meals can be prepared by doing the following sub-tasks in order $R\{drop\ ingredientsin\ a\ pot,\ drop\ cooked\ meal\ on\ plate,\ drop\ mealat\ serving\ counter\}$.

Delivering a meal will add to the total score which is shared between the players. As multiple tasks can be done simultaneously cooperation between the players is necessary to improve the score. The layout of the kitchen can change the dynamic of the game by limiting the access to pots, plates or ingredients for a player creating greater dependencies between the players.

4.2 Training setup

For training the agents I used the principles of SyKLRBR making use of already existing methods from the Overcooked environment [3]. I used 6 agents to create the hierarchical structure as described in 3.2. This was a choice between performance and training time, I opted for a fixed number to create fewer variations during testing. At the start of every epoch, each level π_k picks a training partner lower in the SyKLRBR hierarchy. The training couples train against each other via the proxy policy optimisation (PPO) [10] training method from the Overcooked environment[3]. In PPO training the couple plays 15 synchronous games of Overcooked with 400 actions per agent. Out of these games, sequences of steps get randomly picked for agent modelling. If a sequence of steps got rewards the model of the agent gets updated to enforce such behaviour. After all agents have played and are updated they receive a new training partner resulting in synchronous training. In the final evaluation, the agents only get evaluated on the number of meals they delivered. Since delivering a meal can be challenging enough for untrained agents small rewards for all the sub-tasks in R can be obtained. These dense rewards guide the agent towards the final action in R at the beginning and are scaled down to 0 during the course of the training session, so the final agent is just optimised for delivering meals.

4.3 Evaluation

For evaluating the agent models the methods self-play and ad-hoc play were tested in different layouts. Self-play tests function as a baseline for the evaluation as there is no influence by other agents. A large difference between SP and ad-hoc play performance would suggest that the agent is overfitted to its training data. To access the potential of SyKLRBR in human-AI coordination cross-play in ZSC [5] settings will be performed. The partners in cross-play are generated via the various existing training methods population-based training (PBT), PPO, and behavioural cloning which were present in the Overcooked environment [3]. These methods were chosen since they have been proven to be useful and existent in the Overcooked environment via the original human-AI research in Overcooked. The Overcooked repository contains data generated by real humans playing Overcooked, which is used to create a human-proxy model via behavioural cloning. All of the agents were trained over 5 million steps in the environment in order to keep comparisons equal. The performance is determined by playing 40 games between a match-up with different seeds and taking the average amount of points gained during a game. The tests will be performed on 2 layouts shown in figure 1. The cramped room will be used to evaluate the low-level coordination like collisions. The forced coordination layout will see if the agent is able to generate a higher-level strategy when neither agent can serve a dish on their own.

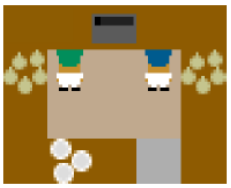
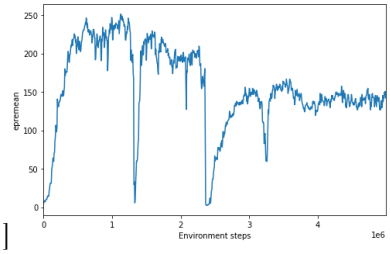
[Cramped room]  [Forced cooperation] 

Figure 1: Layouts used during Evaluation

5 Results and discussion

5.1 Training results

After 5 million steps per agents all SyKLRBR agents have been able to converge towards a successful strategy in the cramped room layout(figure 1). Agent 1 from figure 2 is trained against the uniform random agent at π_0 as no other agents are below it. Although it's partner is not very cooperative it manages to rapidly grow to a strategy which averages out above 200 points per game. As the dense rewards decrease the average rewards decreases to with a big spike downwards. We hypothesise that the agent overfits to the sparse reward gained with help of the random agent, creating a method which is not reliable. After 3 millions steps the dense rewards are completely gone, agent 1 has found a reliable strategy where it probably leaves out any cooperation with level 0. Agent 5's learning curve shown in figure 2 shows a lot more variance due to the 5 different agents is plays. The learning curve of π_5 shows resemblance to π_1 by downward trends at the points where level 1 drops down, indicating that strategies from below to transcend all the way up to the top. The last 2 millions steps of agent 5 shows it has been able to generate a strategy for all agents between π_4 and π_1 by consistently getting a average score just below 150. The significant drops resemble the games played against π_0 whom couldn't be fitted into the general strategy as its behaviour is to different.

$[\pi_1]$ 

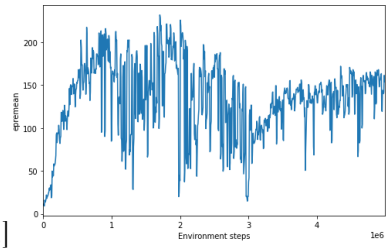
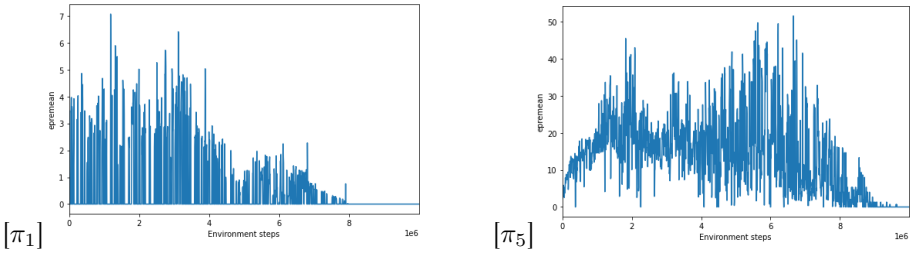
$[\pi_5]$ 

Figure 2: Learning curve of SyKLRBR on the cramped room

Where SyKLRBR was able to find a reliable strategy in the cramped room (figure 1) it did perform poorly in the forced coordination layout (figure 1). Although π_1 is learning at first, a consistent decline is forced by the decreasing dense rewards as shown in figure 3 In the forced coordination layout the agents are reliant on each other in order to deliver a meal. The first layer only trains against the the random agent giving a return of 1/2000

delivered meals per game on average. This makes it impossible for π_1 to create a strategy that is optimised for the dense reward. The agents higher up in the hierarchy are able to develop strategy that is able to collect sparse rewards but slowly regress after the dense reward horizon by trying to optimise their strategy towards π_1 andπ_0.



$[\pi_1]$ $[\pi_5]$

Figure 3: Learning curve of SyKLRBR on the forced coordination

5.2 Evaluation results

The model SyKLRBR created for the cramped room played against other Overcooked agents in order to access its performance in ZSC setting [5]. From the cross-play table we can tell that PBT did perform the best in this ad-hoc play. This could be explained by the characteristics of PBT, training against multiple opponents synchronously just as SyKLRBR creates a lot of exploration during training leading more robust agents. PPO, a self-play based agent, does manage to get high scores in ad-hoc play suggesting that indivdual strategies thrive in this small layout. As PBT has much a higher self-play score compared to SyKLRBR we hypothisise that the ability to create a more optimised strategy by not having to train against unoptimised like pi_0 is giving it an edge in ad-hoc play in the cramped room. Noteworthy is the fact that SyKLRBR is much more consistent over all the partners. This suggests that SyKLRBR is capable of generating a strategy that is robust against more strategies compared to PBT.

Training method	SyKLRBR	PBT	PPO	BC	Human-proxy
SyKLRBR	123,34	120,11	90,54	111,59	106,91
PBT	120,11	155	146,79	93,29	89,23
PPO	90,54	146,79	199,19	98,43	100,67
BC	111,59	93,29	98,43	109,55	105,86
Human-proxy	106,91	89,23	100,67	105,86	103,52
Average ad-hoc	107,28	112,355	109,13	102,29	100,66

5.3 Discussion

The big difference between the results of the cramped room and forced coordination layouts is the interdependence that they require. Where pi_1 of the cramped room is able to generate a perfectly optimised strategy within a million steps, the forced coordination failed to even generate a functional strategy. The random agent in SyKLRBR completely eliminates learning progress for the layers directly above it making the first 2 layers close to static. Since every layer chooses its training partner via the Poisson distribution downwards in the hierarchy the training partner distribution becomes $\pi_0 \approx 40\%, \pi_1 \approx 20\%, \pi_2 \approx 17\%, \pi_3 \approx 15\%, \pi_4 \approx 8\%$. When the first 2 layers are not learning you are training against static agents for potentially

60% of the time. To avoid the regression of the higher layers later on in the training as seen at π_5 in figure 3 the partner selection should be adjusted to the environment.

Although SyKLRBR did not outperform any other training method there is potential for it in the human-AI field. PBT and PPO did perform very well in ad-hoc play as their optimised strategies were able to carry them through the games. The algorithms used to generate the models are also more optimised than SyKLRBR. SyKLRBR did not have tuned hyper-parameters and used its last model of training instead of the best. Combined with future improvements I hypothesise that SyKLRBR could easily outperform PBT and PPO in a bit more cooperative setting.

6 Responsible Research

As the paper only dove into the specific training method SyKLRBR the direct impact of this research will be limited. The results of this paper could be useful for further research on the human-AI topic and thus need to be reproducible and generated legitimately. The methodology I used and described is important for other researchers to decide whether to further research SyKLRBR or opt for other training methods. As the algorithm still has limited research data and is not fully developed for a greater scale I do not think companies will start using it. Therefore I foresee a very little chance that this paper will have a direct impact on others.

Since the time and computational resources were limited during this research the amount of generated data is limited, increasing the importance of reproducibility. As a researcher, I hoped to have more data in order to more conclusively determine the causes of the results. Since the data is limited I did incorporate almost everything in the report. The results are thus not cherry-picked and represent the actual findings during this research. Data that was left out of the report was obtained by an earlier version of my SyKLRBR implementation. This data did not correctly represent SyKLRBR, and was not found worthy to compare. The results that were obtained are reproducible, as the output of my SyKLRBR implementation were close to deterministic over different seeds. This not only gives confidence in the results but is also useful for future research.

The reproducibility of my experiments are high as all source code and data is available on this GitHub repository [11]. The data and trained agents used in this paper are stored in the repository for future research or verification of the methods. All code was run on a laptop with a maximum training time of 8 hours per seed making it possible for everyone to reproduce or generate new data. The code built on top of the originally documented repository is described in detail in section 4 which should be used as a guide to understand the code and training process.

7 Conclusions and Future Work

The research started with the question **"Does an agent trained with SyKLRBR perform well in ad-hoc play in Overcooked?"**. To answer this question I've implemented the algorithm into Overcooked and compared it to current AI training methods. While this research has been performed under limited time I can conclude that SyKLRBR isn't perfect but does hold potential in ZSC settings. SyKLRBR does not perform well in highly cooperative settings as learning is limited by the static uniform random agent. Although it helps

to explore the environment at the beginning of training, it later on becomes a burden that can prevent any form of cooperation. The cross-play tests showed that SyKLRBR was able to generalize very well over different strategies at test time, but was not able to optimise its strategy because a large amount of training against the lower levels of the hierarchy. Where SyKLRBR was tested in low and high cooperative environments I foresee its best use somewhere in between those 2.

7.1 Future work

Limited time and computing power during the research have left some unanswered questions and also created new ones. The Overcooked environment still offers lots of room for exploration on SyKLRBR. In both of the environments as SyKLRBR many parameters were fixed in order to keep the results comparable. In the environments I've limited myself to 2 completely different layouts, one where cooperation was unnecessary and one where it was forced. Future research could discover the limits of cooperation for SyKLRBR by using different layouts. The SyKLRBR implementation was kept close to the description in the Hanabi paper [5], which leaves room for further improvements of SyKLRBR in Overcooked. Different sizes of the hierarchy or the distributions used for picking training partners could be potential improvements. These parameter improvements could improve the adaptation to more cooperative layouts or the Overcooked environment as a whole. A bigger addition could be the use of a recurrent policy like LSTM networks, this would give the agents the ability to recognize their partner's strategy at test time. A broader selection of strategies could be more functional in ad-hoc play compared to one generalised strategy. With the ability to recognise strategies research into the use of lower layers could be interesting as a bad player could potentially perform better with an agent which is more optimised against random agents.

References

[1] M. Carroll, R. Shah, M. K. Ho, *et al.*, "On the utility of learning about humans for human-ai coordination", *Advances in neural information processing systems*, vol. 32, pp. 5175–5186, 2019.

[2] B. Cui, H. Hu, L. Pineda, and J. Foerster, "K-level reasoning for zero-shot coordination in hanabi", in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 8215–8228.

[3] M. Carroll, *Humancomputable-ai*, `https://github.com/HumanCompatibleAI/overcooked_ai`, 2019.

[4] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments", *Advances in neural information processing systems*, vol. 30, pp. 6382–6393, 2017.

[5] H. Hu, A. Lerer, A. Peysakhovich, and J. Foerster, "Âother-playâ for zero-shot coordination", in *International Conference on Machine Learning*, PMLR, vol. 2, 2020, pp. 4399–4410.

[6] H. Hu, A. Lerer, B. Cui, L. Pineda, N. Brown, and J. Foerster, "Off-belief learning", in *International Conference on Machine Learning*, PMLR, 2021, pp. 4369–4379.

[7] D. Strouse, K. McKee, M. Botvinick, E. Hughes, and R. Everett, "Collaborating with humans without human data", M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, 2021, pp. 14 502–14 515.

[8] D. Shapiro, X. Shi, and A. Zillante, "Level-k reasoning in a generalized beauty contest", *Games and Economic Behavior*, vol. 86, pp. 308–329, 2014.

[9] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks", *Chemometrics and Intelligent Laboratory Systems*, vol. 39, no. 1, pp. 43–62, 1997.

[10] Y. Wang, H. He, and X. Tan, "Truly proximal policy optimization", in *Uncertainty in Artificial Intelligence*, PMLR, 2020, pp. 113–122.

[11] J. Vos, *Syklrbr-repo*, `https://github.com/Jimvos32/SyKLRBR`, 2022.