Coupled and Model-based cooperative planning in Overcooked AI

Nils van Veen
Supervisor(s): 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

In the field of cooperative AI, an environment is created called Overcooked AI based on the popular Overcooked game. Originally the environment is used to study deep reinforcement learning, on the other hand it also allows for cooperative planning methods of which the paper will focus on. These methods include coupled based planning with replanning and model-based planning. This research paper attempts to reproduce the results the Overcooked AI environment developers obtained and to improve the Coupled Planning algorithm to gain higher results. In particular, experiments were performed against themselves and a human model for the planning methods, and an improved coupled planning algorithm, in which the failures are handled by deviating from optimal play, under different game steps. And a study on collision failures is performed. The results concluded that extrapolation of results are sub-optimal and that collision failures can be significantly reduced by handling collision differently; walking into the opposite direction.

1 Introduction

Overcooked [1] is a cross-platform multiplayer game in which up to four players work together in a kitchen to prepare and serve meals. The collaborative nature of the game and the various range of cooking tasks players could be asked to complete, makes Overcooked an ideal environment for studying human-AI cooperation and ad-hoc teamwork. A simplified version of the game is made called Overcooked AI [2]. While the Overcooked AI environments' main intend was to study reinforcement learning for human-AI cooperation, it does not take away that state transitions can be used to make (cooperative) planning based agents feasible as well. The challenge here, is to plan without knowing how the other agent behaves and what plan that agent will follow.

The research can be mainly split into two different approaches to the collaborative planning problem: (1) Coupled planning with replanning, where the AI computes an (close-to) optimal joint for both players from the current state, and re-plans when the other player takes an unexpected action. And (2) model-based planning, where the AI computes an optimal plan for its own actions, using a learned human model to predict the actions the human would take in each state. Even though, both these methods contribute towards this paper, the focus will be on coupled planning with replanning as model-based planning proved to be too computationally intensive to evaluate, given the computing resources. This leads to the following research questions:

1. What are the strengths and weaknesses of coupled planning with replanning as a solution to the ad-hoc teamwork problem?

2. What can be improved upon in the current coupled based planning with replanning approach in [3]?

In this paper, it is found that coupled planning with replanning performs extremely well in self-play due to its (close-to) optimal play. When paired with a human agent it has trouble adapting to the human, due to non-optimal play. The fact that humans play slower and hence might use possible incompatible strategies also contribute towards this. The performances obtained are in line with the original paper [3]. There are however differences to the original results due to extrapolation and seed dependencies. By improving collision failures, the coupled planning can be significantly improved and the amount of failures will have a neglectable impact on the final outcome.

The remainder of the paper is structured as follows; First of all, the background of the research. Secondly, the Methodology will be discussed on the approach of solving and tackling the research questions. Thirdly, the results will be provided and discussed. Fourthly, a chapter on Responsible Research to go over the ethical aspects of the research and in particular the repeatability of the methods. The last chapter summarizes the main conclusions and an overview of future work is provided.

2 Related work

2.1 Overcooked AI

In this paper, the *original Overcooked AI paper* is a reference to [3]. The paper evaluates agents trained in the Overcooked AI environment [2]. This includes planning based agents and agents trained via self-play and population-based. Evaluation was done in several different approaches to multi-agent reinforcement learning, based on the PPO [4] algorithm. Scenarios were created to test different coordination problems, including: 1) collision avoidance, 2) coordination where there are multiple incompatible solutions, and 3) sub-task allocation based on each agent's individual capabilities. Key findings of the paper describe that adaptivity to humans and being able to lead as well as follow are important for higher overall reward, instead of expecting optimal-play from the partner-agent. Figure 5 and Appendices E, F and G from the paper [3] are dedicated to planning experiments. Due to the nature of planning algorithms used in the paper, all agents choose actions deterministically. This lead to collision failures resulting in getting stuck often - lasting the whole remainder of the trajectory - having little or no reward as a result. Overall, the paper concluded that agents trained using a combination of self-play and play against human models were the most robust in cross-play, and did better when paired with real human players.

2.2 Multi-agent Planning

[5] Describes how high-level and low-level action plans are coordinated by inverse planning for cooking tasks, such as the ones in Overcooked. It also describes how human judgements can be reflected w.r.t. the prediction of sub-task allocation. [6] Goes into more detail on how human-aware task and motion planning work, as well as giving more insight into interaction schemes considering human abilities and preferences. [7] Analyses how environments in Overcooked result in different behaviour, while using the same planning algorithms. [8] Describes how non-experts (humans) can solve and avoid failures in high-level actions. [9] Introduces a model-based planner which can learn to construct, evaluate and execute plans. [10] Provides guidance through the foundation of motion planning.

2.3 Ad-hoc Teamwork

In [11], the effects of communicating at a cost in planning w.r.t. ad-hoc teamwork is discussed. [12] Explains how human behaviour can impact ad-hoc teamwork, under the authors experiments. In [13] planning in ad-hoc teams is tackled as an optimization problem in joint policy space, where the agent's policies are evaluated via Monte Carlo simulation. Ad-hoc teamwork under partial observability is tested in Overcooked AI, where a novel Bayesian online prediction algorithm is used to tackle Overcooked as an unknown task with unknown

teammates [14]. The paper [14] also reasons about dividing roles under agents, specifically a cook and helper role.

3 Background

Two types of planning algorithms are used in the original paper [3]; (1) *coupled planning with re-planning* (CP) and (2) *model-based planning* to work in the simplified Overcooked environment. These planning algorithms are evaluated on two different overcooked scenarios: Cramped room and Asymmetric advantages.

3.1 Coupled Planning with Replanning

For coupled planning, a near-optimal joint plan is computed for both agents. While progressing through a level step-by-step, agent X executes its move and agent Y does too. If agent Y deviates from the expected 'optimal' move, then agent X re-plans the entire near-optimal joint plan. Note that agent X is a planning-based agent, while Y is *some* other agent.

Near-optimal joint motion plans are pre-computed for every possible initial and desired final state of an agent. This is computed by creating a search tree where the initial node is the current state and where the child nodes, recursively, are the successor states from a joint-action executed by a player (e.g. by walking into a direction). A path is considered a motion when it reaches a goal state. A goal state is a state in which a task is done, e.g. a dish delivered. To calculate the cost for each motion, all specific joint motion plans are mapped to some high-level action. An example of a high-level action is: "pick a plate". Then, an A^* search is used on this high-level action space to find the joint plan and its respective A^* search cost. The default heuristic used to calculate the A^* cost from Overcooked AI [2] involves calculating the following costs: (1) pot to delivery, (2) dish to pot and (3) item to pot. The combination of these costs is taken as the heuristic cost.

The planner does make some simplified assumptions, making it near-optimal instead of optimal, such as: (1) When no plan is found, the agent will randomly walk into a direction, and (2) To reduce computations by a factor of sixteen, the orientations are ignored. As these decisions bring side-effects, they will be discussed in further detail in the Methodology, and Results and Discussion sections.

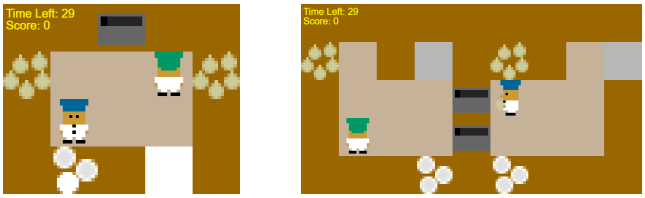
3.2 Model-based Planning

For Model-based planning, the planner is given access to a human model. The human model used in the experiments are (1) Behavioural Cloning (BC) and (2) the actual (true) human, which is referred to as the Human Proxy a.k.a. H_{Proxy}. Behavioural cloning [3] is also considered imitation learning; it learns some policy from demonstration and maps observations to actions directly with standard supervised learning methods [3]. The agent uses an A^* search to find a near-optimal action based on a model of the other agent (BC or H_{Proxy}) and decides its own action. The A^* search consists of two layers: (1) On a low level, in actual game state space, where edges in the graph are player action joints. To reduce computation cost, the developer decided to remove stochasticity by taking the class with largest predicted probability (i.e. argmax probability action) [3]. (2) On a high level, the edges are high-level actions. This is similar to the near-optimal joint planner used in the Coupled Planner. The only exception is that, in the Model-based planner it is unfeasible

to pre-compute all low-level motion plans [3], due to too many dependencies; start state, end state, orientations and other state features. These factors could influence the actions returned by the partner model.

3.3 Overcooked Scenarios

Due the computational complexity, the model-based planner was only feasible on two layouts; Cramped room and Asymmetric Advantages, see Figure 1. Hence for this research the focus will be on these two layouts. Cramped room is a small room in which the agents easily collide with each other, while Asymmetric Advantages tests whether agents can maximize their own strength by developing high-level strategies.



(a) Cramped Room (b) Asymmetric Advantages

Figure 1: Two layouts from Overcooked AI: Cramped Room and Asymmetric Advantages

4 Methodology

Reproducing results from the original Overcooked paper [3] is an important part of this research as it is a starting point for analysis on collision failure cases and on possible improvements. The following sections describe what is done to reproduce the results.

4.1 Configurations

Firstly, setting up the existing Overcooked AI environment is done to make sure all required dependencies from [15] work well together. After this, different configurations of agents are evaluated against each other. The agents used are: (1) *CoupledPlanningAgent*, this agent type makes use of a coupled planning approach. (2) *EmbeddedPlanningAgent*, makes use of a model-based planning approach. (3) *HumanProxyAgent*, this agent type is the Human Proxy.

The configurations of planning agents used in the original paper [3] are: self-play of CoupledPlanningAgent ($CP+CP$), CoupledPlanningAgent and HumanProxyAgent ($CP+H_{Proxy}$), EmbeddedPlanningAgent and HumanProxyAgent: (1) the model used is the behavioural cloned human ($P_{BC} + H_{Proxy}$) and (2) the model used is the human proxy ($P_{H_{Proxy}} + H_{Proxy}$). For the non-self-play configurations, the agents also ran on switched indices, e.g. ($H_{Proxy} + CP$).

4.2 Evaluation

In the original paper [3] the results listed are based on a 400 step horizon. To easily compare results to the original paper [3], it is decided to do the same in this paper. One step

is a player action: Walk north/east/south/west, stay or interact. The Model-based planning experiments proved to be too computationally intensive to evaluate with the available computing resources, hence these experiments were only run once for all the configurations on 400 steps to get results to compare with. The main focus is on the coupled-planning experiments.

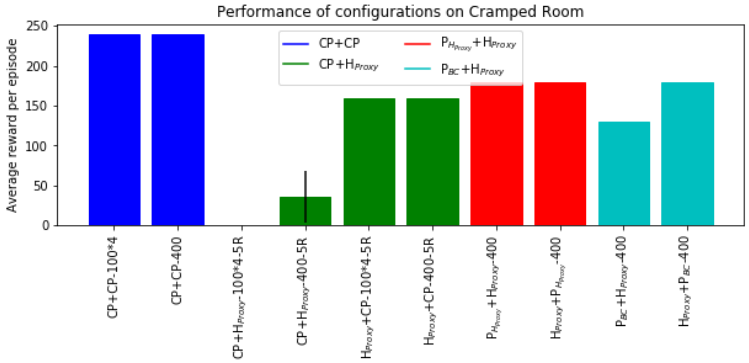
The coupled planning agent pairs are evaluated on four types of different configurations. As Self-play had consistent results, the configurations used are: (1) once, on 100 steps, (2) once, on 400 steps. Non-self-play was evaluated under the following configurations: (1) average of five times, on 100 steps, (2) average of five times, on 400 steps. One of the reasons for these different types of configurations is to test whether the results from the original paper [3] are sub-optimal and if so, by how much approximately. In the paper [3], it is mentioned that the experiments were run on a step horizon of 100 and extrapolated (multiplied by four) to get the result of a 400 step horizon. Another reason is to have clear results to compare to with potential new agents which are deviations from a planning agent. The main criteria for evaluation is the average reward per episode (of 400 steps). Additionally, the amount of failures caused by collisions during planning is saved to compare with new agents, to find a correlation between reward-improvement and failures. To stay consistent over all runs, seed 166 was used.

4.3 Improved Coupled planner

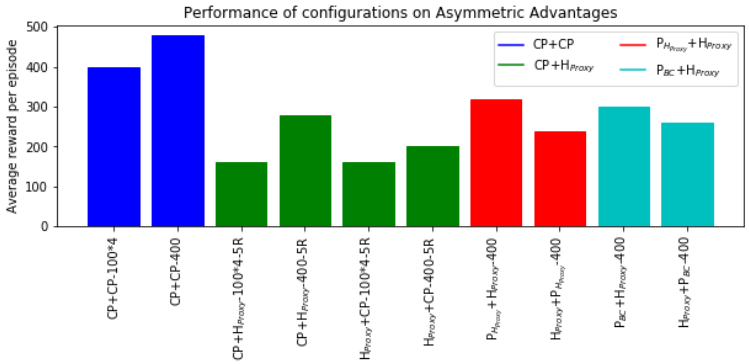
To test adjustments to the coupled-based planner, a CoupledPlanningExperimentalAgent (CP_X) agent is made. This agent makes use of the same logic as the CoupledPlanningAgent, however instead of going into a random direction when no joint action plan can be found, it will go in the opposite direction of its current direction. As the primary focus is on improving the planner to handle human behaviour better, a hypothesis is that *if the coupled planner backs off for a move to let the human walk further, then the conflict will be solved and only a couple time steps will be wasted instead of trying the same move a couple of steps after each other and therefore waste even more steps, i.e. expected is to have fewer conflicts with this strategy compared to the original resulting in a higher outcome reward.* To keep track of the coupled planning failures, a counter is added to the evaluated agents which is increased every time there is a Timeout-Error in [2].

5 Results and Discussion

All planning experiments related to reproducing the results from [3], can be found in Figure 2. Full-size result figures (for all experiments together) can be found in Appendix A.



(a) Average reward comparison with planning configurations on Cramped Room.



(b) Average reward comparison with planning configurations on Asymmetric Advantages.

Figure 2: (a) Performance on Cramped Room. (b) Performance on Asymmetric Advantages. x-axis should be read as: planning configuration - steps∗multiplication - runs R. Higher *Average reward per episode* is better.

5.1 Comparison original paper

Compared to the original paper [3], the obtained values for Coupled Planning are close to each other. Reproduced results are listed in Figure 2. The results from the original paper [3] can be seen in Figure 3.

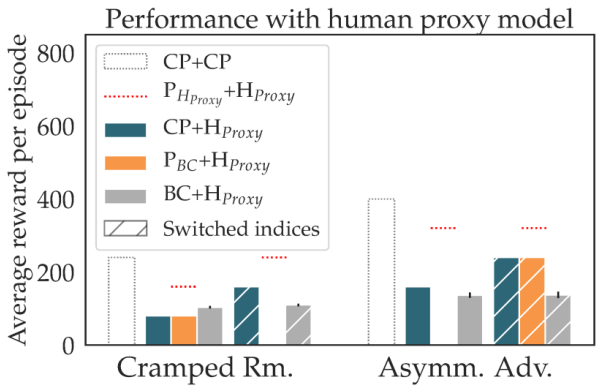


Figure 3: Comparison across planning methods. Reproduced from the original Overcooked paper [3].

Extrapolation is the main reason for this difference. Also, the fact that the results in the original paper [3] are run using unknown seeds, is most likely a source of difference. Self-play ($CP + CP$) performs extremely good, as expected due to (close-to) optimal play. When paired with a human agent ($CP + H_{Proxy}$) the performance drops and collision failures happen, this is also expected due to low adaptivity to the human agent. Performances are in line with the original paper [3].

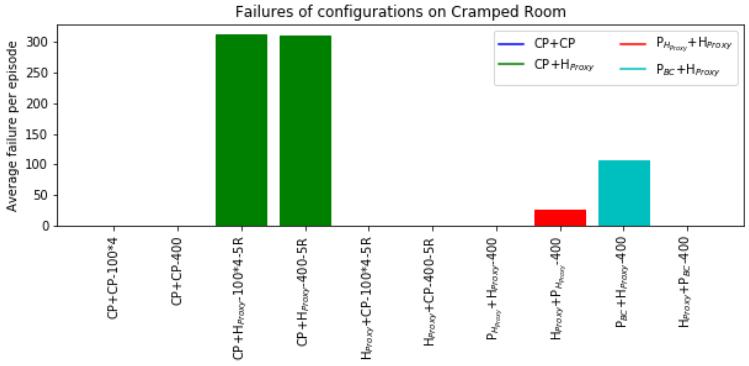
For model-based planning, the results of $P_{BC} + H_{Proxy}$ and $P_{H_{Proxy}} + H_{Proxy}$ are close to each other, if this is due to seed randomness, is still to be investigated.

5.2 Extrapolation Suboptimality

According to the original paper [3], planning experiments are evaluated on a horizon of 100 steps and then multiplied by four to make the environment horizon comparable to the other experiments. This is possibly a source of suboptimality. As this is not tested in the original paper, it was decided to test it in this paper. As can be seen in Figure 2a and 2b, it is not always sub-optimal. However, in most cases it is sub-optimal. This can be easily seen by comparing a configuration ran on 100 steps multiplied by four to a configuration ran on 400 steps. E.g. $CP + H_{Proxy} - 100 * 4 - 5R$ vs $CP + H_{Proxy} - 400 - 5R$ on Asymmetric Advantages. Here it can be seen that the average over five runs for both configurations yield different results and that 400 steps clearly ends up with higher reward. The same reasoning can be applied on other $100 * 4$ vs 400 steps configurations where the outcome is lower for extrapolated 100 steps.

The suspected reason of this result is that in between 0 and 100 steps of doing a task, which is abruptly quit at 100 steps, some steps are not actually reflected in the result. At the moment of quitting, the agents are still doing a task and the steps taken towards solving the dish will never be reflected in the result of the 100 steps, however they will be reflected if it is stopped at 400 steps. A side effect here is that, as some steps towards 100 steps, are four times not used due to extrapolation. Those neglected steps could be the source of difference between extrapolated 100 compared to 400.

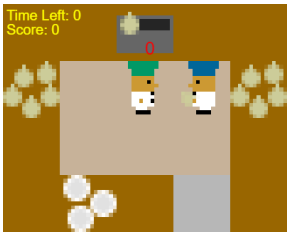
5.3 Collision Failures



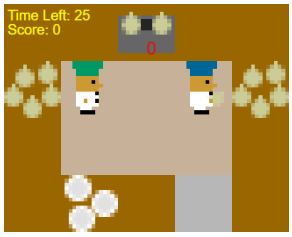
(a) Average collision failure comparison with planning configurations on Cramped Room.

Figure 4: (a) Collision failures on Cramped Room. x-axis should be read as: planning configuration - steps∗multiplication - runs R. Lower *Average failure per episode* is better.

Collisions do not occur in Asymmetric Advantages and hence no collision failures happen on this layout. For planning configurations on collision-sensitive layouts such as Cramped Room, agents get stuck often in loops that lasts the whole remaining part of the trajectory, leading to little or no reward [3]. This can be clearly seen in Figure 2, in particular $CP + H_{Proxy}$ shows it well. For this configuration, Figure 2a shows that most of the time no reward is obtained, with the average of five runs on 400 steps being an exception. On the other hand this exception has an extreme standard error leading to inconsistent results. Figure 4a shows that for the same configurations the amount of failures is directly correlated to the reward in Figure 2a, i.e. high amount of collision failures (Figure 4) yield lower reward per episode (Figure 2). From observing when collision failures happened, it is found that most of the collisions are caused by blocking or impossible moves, as seen in Figure 5. A possible reason for this human behaviour is that they are slower and that they want to perform incompatible strategies, besides non-optimal play.



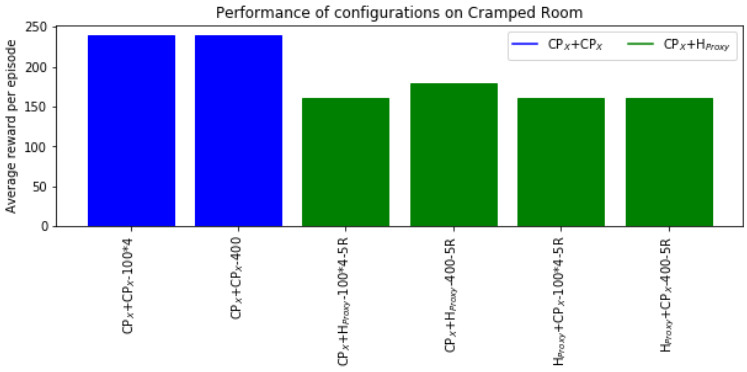
(a) Collision failure case 1.



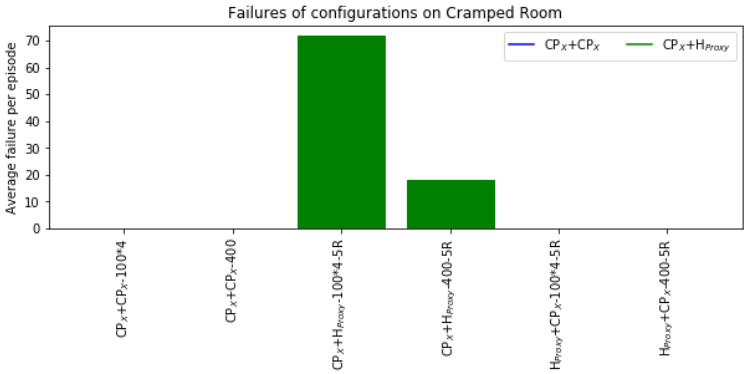
(b) Collision failure case 2.

Figure 5: In (a) the agents try to run into each other, either one of them or both block each other. In (b) the agents try to move to the space in-between them, which is impossible at the same time.

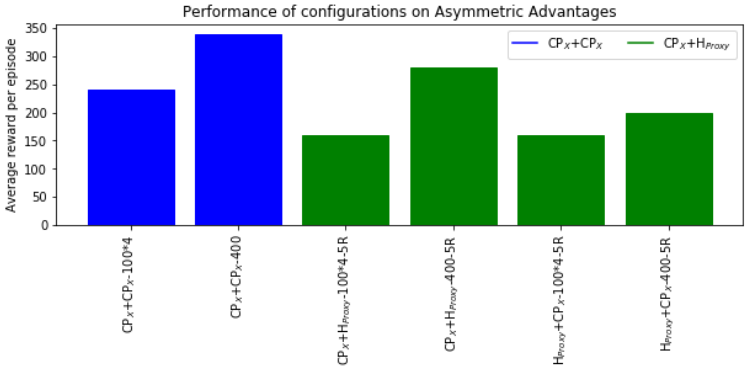
5.4 Reduction Collision Failures



(a) Average reward comparison with planning configurations on Cramped Room.



(b) Average collision failure comparison with planning configurations on Cramped Room.



(c) Average reward comparison with planning configurations on Asymmetric Advantages.

Figure 6: (a) Performance on Cramped Room. (b) Collision failures on Cramped Room. (c) Performance on Asymmetric Advantages. x-axis should be read as: planning configuration - steps∗multiplication - runs R. Higher *Average reward per episode* and lower *Average failure per episode* is better.

The hypothesis stated in the Improved Coupled Planner section [4.3] is confirmed in Figure 6. CP_X clearly shows a significant reduction in failure cases and a significant improvement in average reward compared to CP, when paired against H_{Proxy}. E.g. by looking at $CP + H_{Proxy} - 400 - 5R$ collision failures are reduced from 310 to 18 and the performance increased from 36 (s.e. 32.20) to 180 (s.e. 0).

The only change is how collisions are tackled. Deviating from optimal play from the coupled planner perspective and letting the human work, solves conflicts efficiently. Not all collisions are resolved, however the amount and duration of collisions are small and in current layouts, it does not seem to have a negative impact on the average reward outcome. CP_X does not change anything in Asymmetric Advantages, which is expected as there are no failures. The only difference is the outcome of self-play, however this is most likely due to seed-randomness. According to [8], the majority of humans from their experiments was able to understand failure cases and act accordingly, this confirms the fact that most of the collisions are solved by the human player, but not all of them.

6 Responsible Research

The research from this paper is reproducible as the agents used and evaluated can be found in the NEURIPS2019 branch [2][15] from the Overcooked AI repository [2]. As well as the Jupyter notebooks used to reproduce and create the results found in the paper, are pushed on a publicly-available GitHub repository[1]. Default parameters are used for all the experiments. CoupledPlanningExperimentalAgent is built on top of the Overcooked AI library, to create this agent, one can follow the steps mentioned in the paper, or follow the steps on the Github repository[2] to use the exact same code and/or inspect the output obtained. To allow for the experiments to be reproducible and verifiable, most randomness is made consistent by using seed 166. The seed setter method used to create the result of the original paper [3] for non-planning experiments is used for the planning experiments in this paper. Note that not all randomness is gone by hard-coding the seed, therefore during the experiments the average over multiple runs is taken to minimize fluctuations.

7 Conclusion and Future Work

The research on Coupled Planning with replanning from this paper has shown that Coupled based planning performs extremely well in optimal-play situations, e.g. self-play. When paired with a human agent, it performs inconsistent on cramped scenarios, where collisions occur frequently. After every collision a new plan is calculated, however this also expects the human to cooperate optimally in the plan, while this is not always the case.

Therefore, an improved Coupled Planning agent is tested, to incorporate human behaviour better as well as adapt more to the human. This is done by changing the way coupled failures, caused by collisions, are handled. By deviating from optimal play, the amount of failures drop significantly and the average reward per episode increases up to an accepted standard along with other configurations.

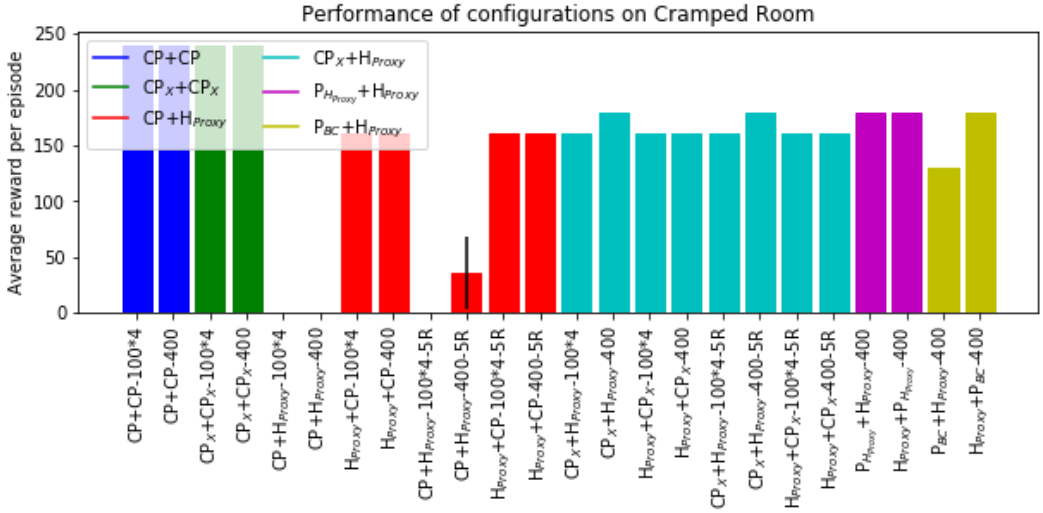
In the original paper [3], sources of sub-optimality in the planning evaluations, specifically coupled planning with replanning, are: Collision failures, possible seed-dependencies and extrapolation from 100 to 400 steps instead of evaluating on 400 steps.

[1] https://github.com/HUILIANGAA/CSE3500_TUD_Planning_Overcooked_AI
[2] See footnote 1.

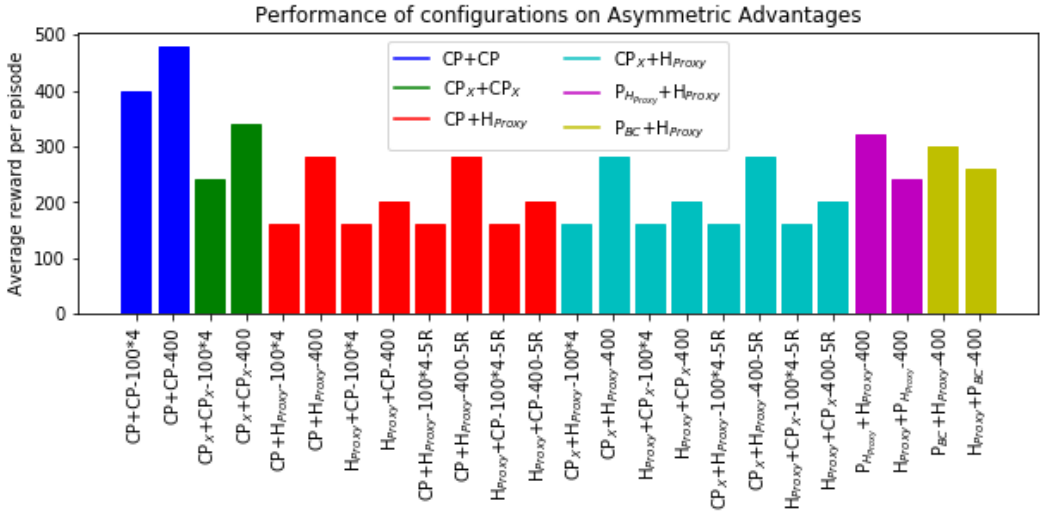
As not all coupled failures are resolved by walking into the opposite direction, the result is still sub-optimal for coupled planning. Therefore, for future work, it is suggested to add position states and orientations of the agents. This will bring more complexity towards the algorithm and will make evaluation significantly slower. It would however be valuable to see whether taken orientations of both agents would minimize collisions and handle collisions better even before they occur. As, the focus of this paper is on coupled planning, the same assumptions and adjustments could be tested on model-based planning.

Additionally, combining ATPO [14] with the current coupled planning algorithm, especially during coupled failures, could be looked into for possible performance improvements. According to analysis in the paper [14], in the Overcooked environment, a random policy (like walking into a random direction) performed significantly worse than ATPO. Also, switching roles from cook to helper during coupled failures could reduce collision failures. On top of that, an algorithm to predict whether the human understands their role in the collision in combination with the statistics of [8] could shine more light on whether the collision will be solved by the human or not and what action should be taken as a result of that to reduce the duration of a collision.

A Full-size Results

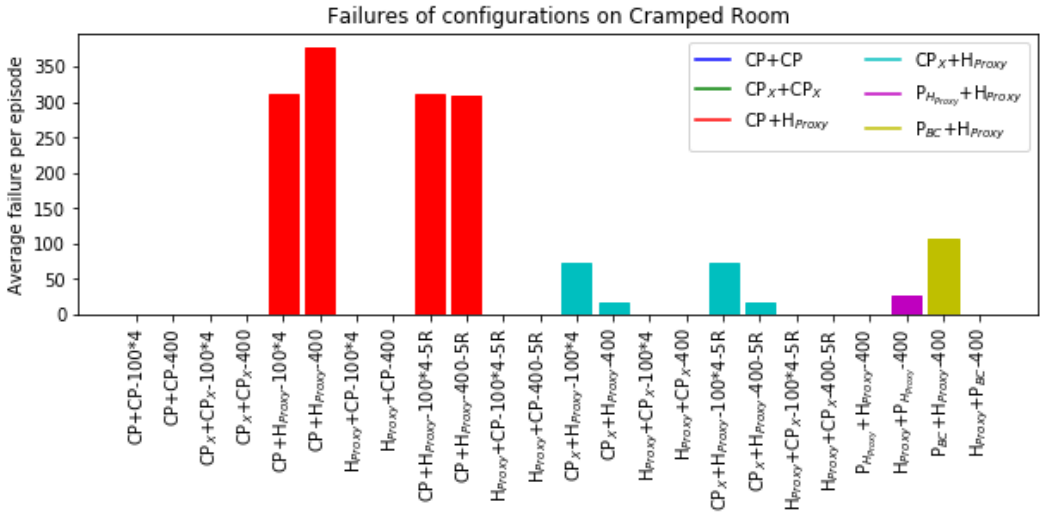


(a) Average reward comparison with planning configurations on Cramped Room.

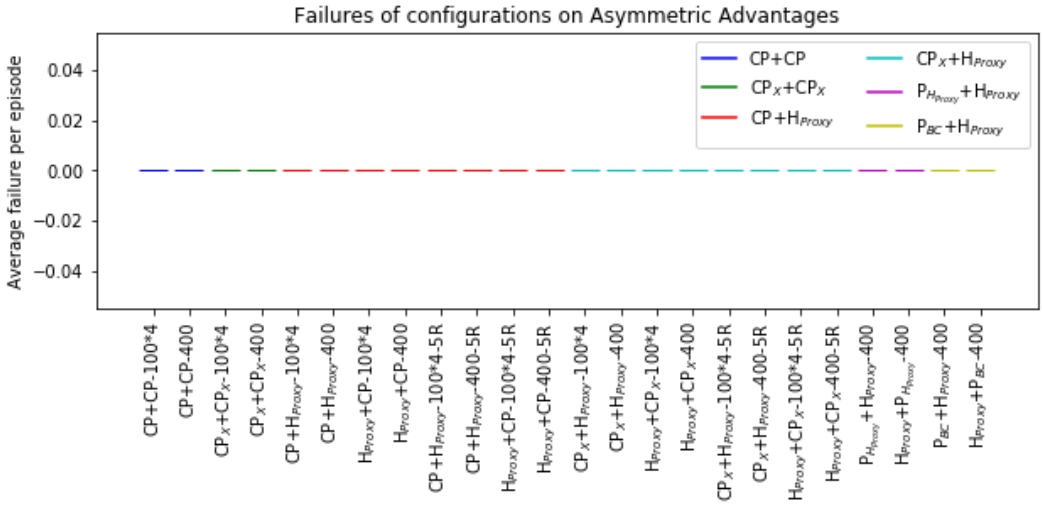


(b) Average reward comparison with planning configurations on Asymmetric Advantages.

Figure 7: (a) Performance on Cramped Room. (b) Performance on Asymmetric Advantages. x-axis should be read as: planning configuration - steps∗multiplication - runs R. Higher *Average reward per episode* is better.



(a) Average collision failure comparison with planning configurations on Cramped Room.



(b) Average collision failure comparison with planning configurations on Asymmetric Advantages.

Figure 8: (a) Collision failures on Cramped Room. (b) Collision failures on Asymmetric Advantages. x-axis should be read as: planning configuration - steps∗multiplication - runs R. Lower *Average failure per episode* is better.

References

[1] G. T. Games, *Overcooked*, `https://ghosttowngames.com/overcooked/`, Published: 2016.

[2] *Overcooked AI environment*, `https://github.com/HumanCompatibleAI/overcooked\_ai`.

[3] M. Carroll, R. Shah, M. K. Ho, *et al.*, "On the utility of learning about humans for human-ai coordination," *CoRR*, vol. abs/1910.05789, 2019. arXiv: 1910.05789. [Online]. Available: http://arxiv.org/abs/1910.05789.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. arXiv: 1707.06347. [Online]. Available: http://arxiv.org/abs/1707.06347.

[5] R. E. Wang, S. A. Wu, J. A. Evans, J. B. Tenenbaum, D. C. Parkes, and M. Kleiman-Weiner, "Too many cooks: Coordinating multi-agent collaboration through inverse planning," in *AAMAS*, 2020, pp. 2032–2034. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/3398761.3399065.

[6] R. Alami, A. Clodic, V. Montreuil, E. Sisbot, and R. Chatila, "Toward human-aware robot task planning.," Jan. 2006, pp. 39–46.

[7] M. C. Fontaine, Y. Hsu, Y. Zhang, B. Tjanaka, and S. Nikolaidis, "On the importance of environments in human-robot coordination," *CoRR*, vol. abs/2106.10853, 2021. arXiv: 2106.10853. [Online]. Available: https://arxiv.org/abs/2106.10853.

[8] S. van Waveren, C. Pek, J. Tumova, and I. Leite, "Correct me if i'm wrong: Using non-experts to repair reinforcement learning policies," Mar. 2022, p. 2022.

[9] R. Pascanu, Y. Li, O. Vinyals, *et al.*, "Learning model-based planning from scratch," *CoRR*, vol. abs/1707.06170, 2017. arXiv: 1707.06170. [Online]. Available: http://arxiv.org/abs/1707.06170.

[10] A. Le and T. Le, "Search-based planning and replanning in robotics and autonomous systems," in Sep. 2018, ISBN: 978-1-78923-578-4. DOI: 10.5772/intechopen.71663.

[11] W. Macke, R. Mirsky, and P. Stone, "Expected value of communication for planning in ad hoc teamwork," *CoRR*, vol. abs/2103.01171, 2021. arXiv: 2103.01171. [Online]. Available: https://arxiv.org/abs/2103.01171.

[12] J. Suriadinata, W. Macke, R. Mirsky, and P. Stone, "Reasoning about human behavior in ad hoc teamwork," in *Adaptive and learning Agents Workshop at AAMAS 2021*, London, UK, May 2021.

[13] F. Wu, S. Zilberstein, and X. Chen, "Online planning for ad hoc autonomous agent teams.," Jan. 2011, pp. 439–445. DOI: 10.5591/978-1-57735-516-8/IJCAI11-081.

[14] J. G. Ribeiro, C. Martinho, A. Sardinha, and F. S. Melo, "Assisting unknown teammates in unknown tasks: Ad hoc teamwork under partial observability," *CoRR*, vol. abs/2201.03538, 2022. arXiv: 2201.03538. [Online]. Available: https://arxiv.org/abs/2201.03538.

[15] *Neurips branch, human-aware*, https://github.com/HumanCompatibleAI/human_aware_rl/tree/neurips2019, Published: 2019.