Mastering Zero-Shot Interactions in Cooperative and Competitive Simultaneous Games

Yannik Mahlau [1] **Frederik Schubert** [1] **Bodo Rosenhahn** [1]

Abstract

The combination of self-play and planning has achieved great successes in sequential games, for instance in Chess and Go. However, adapting algorithms such as AlphaZero to simultaneous games poses a new challenge. In these games, missing information about concurrent actions of other agents is a limiting factor as they may select different Nash equilibria or do not play optimally at all. Thus, it is vital to model the behavior of the other agents when interacting with them in simultaneous games. To this end, we propose **Albatross**: **A**lphaZero for **L**earning **B**ounded-rational **A**gents and **T**emperature-based **R**esponse **O**ptimization using **S**imulated **S**elf-play. Albatross learns to play the novel equilibrium concept of a *Smooth Best Response Logit Equilibrium (SBRLE)*, which enables cooperation and competition with agents of any playing strength. We perform an extensive evaluation of Albatross on a set of cooperative and competitive simultaneous perfect-information games. In contrast to AlphaZero, Albatross is able to exploit weak agents in the competitive game of Battlesnake. Additionally, it yields an improvement of 37.6% compared to previous state of the art in the cooperative Overcooked benchmark.

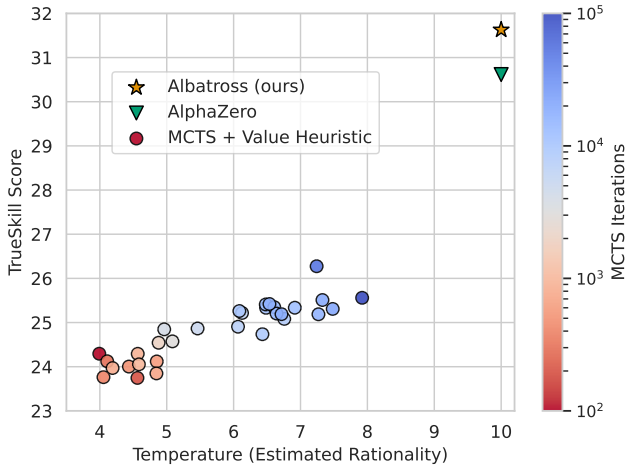


Figure 1. TrueSkill scores (Herbrich et al., 2006) of a tournament consisting of an Albatross agent, Monte-Carlo-Tree-Search (MCTS) baseline agents and an AlphaZero baseline. Each game takes place in a free for all setting of four agents in the stochastic simultaneous game of Battlesnake. Albatross estimates the temperature, i.e. rationality, of the baseline agents online using only data from the current game. A temperature of 0 corresponds to random play and 10 to optimal play if all other agents play optimally as well. AlphaZero achieves optimal play given that all other agents play optimally (temperature of 10), but fails to adapt to subrational agents. In contrast, Albatross is able to respond optimally against any combination of weak and strong agents due to its rationality estimation, resulting in a higher TrueSkill tournament score.

1. Introduction

Games have been played by humans for centuries, some of the earliest dating back more than 4000 years (Sebbane, 2001). They enable us to measure skill, either in cooperation or competition with other agents. When facing unseen agents, one has to adapt to their playing style, which is called zero-shot interaction (Hu et al., 2020). In sequential games, this only entails finding the best response after observing the action of the other agent. However, in simultaneous games, where all agents make their move at the same time, it is also necessary predict the next concurrent move. Therefore, opponent modelling is an important factor for zero-shot interactions in simultaneous games. Most existing methods learn a policy that performs well with or against as many agents as possible (Strouse et al., 2021; Lupu et al., 2021; Zhao et al., 2023; Dockhorn & Kruse, 2017). However, a more scalable approach is the prediction of the other agent's behavior, based on the immediate interactions within a single episode. Lou et al. (2023) apply this idea and classify other agents into groups of low, medium or high skill.

[1]Department for Information Processing, Leibniz University Hannover, Germany. Correspondence to: Yannik Mahlau <mahlau@tnt.uni-hannover.de>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

In contrast, we model their strength as a continuous scalar temperature parameter. The continuous parametrization is able to accurately model any playing strength. Additionally, its sparsity enables efficient maximum likelihood estimation (MLE) using only interactions from the current episode. For this continuous opponent model, we develop the novel concept of a Smooth Best Response Logit equilibrium (SBRLE). Since the SBRLE is intractable in all but very small games, our method **Albatross** (**A**lphaZero for **L**earning **B**ounded-rational **A**gents and **T**emperature-based **R**esponse **O**ptimization using **S**imulated **S**elf-play) learns the equilibrium through a combination of self-play and planning akin to AlphaZero (Silver et al., 2018).

The adaptive behavior of Albatross allows cooperation with unknown agents, which might not act optimally. To this end, we evaluate our method in the game of Overcooked (Carroll et al., 2019), where Albatross is paired with a human behavior cloning agent. Encoding the estimated rationality via a scalar enables us to dissect the different strategies of Albatross when acting with agents of different strengths. In competitive games, the opponent modelling allows Albatross to exploit weaker agents as well as compete with strong agents. In Figure 1, we demonstrate these capabilities in a tournament of Battlesnake (Chung et al., 2020), for which we publish an efficient implementation. Battlesnake is an extension of the well-studied game Tron (Samothrakis et al., 2010; Saverino, 2011; Lanctot et al., 2013; Knegt et al., 2018; Jeon et al., 2022), offering additional stochastic environment dynamics for either two or four agents. In summary, our **contributions** are the following:

- We introduce the novel equilibrium concept of a Smooth Best Response Logit equilibrium for modeling asymmetric bounded rationality with a single rational and an arbitrary number of weak agents.
- Our method Albatross learns to approximate an SBRLE using a mixture of self-play and planning, adapting AlphaZero in a principled way based on game theory to zero-shot interactions of simultaneous games.
- We empirically evaluate Albatross in several cooperative and competitive games and perform an extensive hyperparameter analysis. Additionally, we qualitatively demonstrate the adaptive behavior of Albatross.
- To support reproducibility, all of our code as well as the trained models are open source[1].

2. Related Work

AlphaGo (Silver et al., 2016) was one of the first successful applications of deep reinforcement learning to a complex multi-agent game, namely Go. It used an actor-critic neural network, which was pre-trained on human games and fine-tuned using self-play. The pre-training phase was eliminated in its successor AlphaGoZero (Silver et al., 2016) to avoid learning a suboptimal bias from human play. Independently of AlphaGoZero, a similar system named Expert Iteration was developed for the game of Hex (Anthony et al., 2017). Both AlphaGo and AlphaGoZero exploited knowledge about symmetries of the games, which prevented its application to other games. AlphaZero (Silver et al., 2018) excludes all game-specific knowledge and therefore is applicable to other games, e.g. Go, Chess and Shogi. MuZero (Schrittwieser et al., 2020) did not only learn perfect play, but also the environment dynamics through self-play, which makes it suitable for environments with unknown dynamics. However, for that purpose, MuZero requires much larger compute resources than AlphaZero.

While the mentioned methods are state of the art in sequential competitive games, they do not necessarily work well in zero-shot coordination tasks. That is, because they do not perform well on all game situations, but rather only on game situations that would arise through self-play (Lan et al., 2022). To achieve good coordination capabilities with a lot of different teammates, Fictitious Co-Play (Strouse et al., 2021) trains against past training checkpoints taken at different time points from multiple self-play agents. In contrast, Trajectory Diversity (Lupu et al., 2021) also aims to train a diverse population of agents by regularizing the loss function with Jensen-Shannon Divergence (Lin, 1991). Similarly, Maximum Entropy Population-Based training (MEP) (Zhao et al., 2023) uses population entropy as regularization. Lou et al. (2023) extend MEP in their framework called Policy Ensemble for Context-Aware Zero-Shot Human-AI Coordination (PECAN) by using a randomly weighted policy ensemble given three groups of agents, ranked based on their self-play performance. Hidden-Utility Self-Play (HSP) (Yu et al., 2023) trains with agents maximizing a hidden reward function and achieves population diversity by filtering with an event-based metric. All of the mentioned techniques rely on the idea of training a diverse population to learn to cooperate with as many agents as possible. However, we believe that explicit opponent modelling based on game theory can produce better cooperation capabilities.

Substantial research regarding opponent modeling is conducted in the field of security games and adversarial domains (Nashed & Zilberstein, 2022). Some approaches are based on Variational Autoencoders (Papoudakis & Albrecht, 2020), Switching tables (Everett & Roberts, 2018), model-based approaches using neural networks (Knegt et al., 2018), recursive reasoning (Yu et al., 2022) or expert imitation (Dockhorn et al., 2017). Applications of opponent modeling include real-world situations like wildlife protection against poachers (Fang et al., 2017) or protection of security re-

[1] https://github.com/ymahlau/albatross

		Player 2			
	Action	c		d	
Player 1	a	-4	4	-7	7
	b	-6	6	2	-2

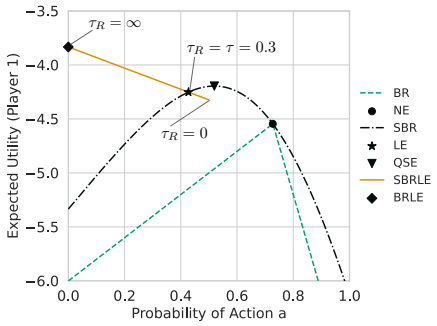


Figure 2. Visualization of equilibria in a zero-sum NFG. Assuming that player 2 plays a best response (BR) to the policy of player 1, the expected utility is lower than under the assumption of a subrational smooth best response $SBR(\cdot, \tau = 0.3)$. The dotted gray lines denote the expected utility of playing actions a or b against an SBR respectively. The NE maximizes the expected utility assuming player 2 plays a BR, while the QSE maximizes under the assumption of an SBR. The SBRLE starts with response temperature τ_R at a uniform distribution over actions a and b, and ends with $\tau_R \to \infty$ at the BRLE. The SBRLE is equal to the Logit equilibrium (LE) if the response temperature τ_R is equal to the temperature τ of the LE.

sources (Yang et al., 2012). However, these methods fail to accurately model behavior in zero-shot interactions, because they require a lot of data to train the opponent model.

3. Game-Theoretic Background

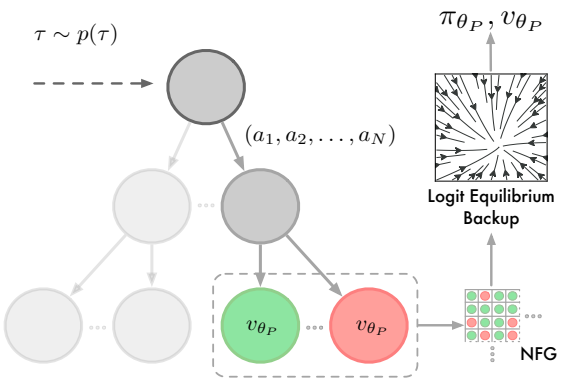
For the game theoretic background, we adapt the notation of Leyton-Brown and Shoham (2008) as well as Milec et al. (2021). A Normal-form game (NFG) is a tuple (n, A, u), where $n \in \mathbb{N}$ denotes the number of agents, $A = A_1 \times \cdots \times A_n$ the joint action set and $u = (u_1, \ldots, u_n)$ the utility functions. We call $a = (a_1, \ldots, a_n) \in A$ a joint action and a_i is the action of agent i. Agents are indexed by $i \in \{1, \ldots, n\}$ and $-i$ denotes the set of all agents except i. We abuse notation and use $-i$ as notation for the other agent in games of two players and the set of all agents except i otherwise. A game of two agents is called zero-sum, if the utility function has the property $u_i(a) = -u_{-i}(a) \, \forall a \in A$. Similarly, a game is fully cooperative if all agents have the same utility function. The set of policies Δ_i (also called mixed strategies) is the set of probability distributions over the action space A_i. A joint policy (also called strategy profile) is a tuple of policies $\pi = (\pi_1, \ldots, \pi_N), \pi_i \in \Delta_i$. Its utility for agent i is defined as the expected outcome $u_i(\pi) = \sum_{a \in A} u_i(a) \prod_{j=1}^{n} \pi_j(a_j)$. The best response (BR) of agent i to the policies of other agents π_{-i} is a policy $\pi_i \in BR(\pi_{-i})$, where the best response function is defined as the set of policies achieving maximum utility: $BR(\pi_{-i}) = \{\pi_i^* \mid u_i(\pi_i^*, \pi_{-i}) \geq u_i(\pi_i, \pi_{-i}) \, \forall \pi_i \in \Delta_i\}$. If all agents play a BR, then their joint policy is called a Nash equilibrium (Nash, 1951).

Nash equilibria inhibit the assumption that all players act rational, which is not realistic in most real-world scenarios. Following Hofbauer and Sandholm (2002), one can incorporate an error probability by transforming the utility functions $\tilde{u}_i(\pi) = u_i(\pi) + \frac{1}{\tau}\psi(\pi_i)$ using Shannon Entropy as a concave smoothing function $\psi(\pi_i) = \sum_{a_i \in A_i} \pi_i(a_i) \log(\pi_i(a_i))$. The temperature parameter τ controls the inverse strength of regularization, i.e. the bounded rationality. Agents can maximize the transformed utilities by playing a smooth best response (SBR), which is a softmax over the original utilities $SBR(\pi_{-i}, \tau) \propto \exp(\tau \, u_i(\, \cdot \, , \pi_{-i}))$. To exemplify the effect of the temperature, a temperature of $\tau = 0$ corresponds to uniform random play and the SBR approaches a BR with $\tau \to \infty$. Akin to BR and Nash equilibria, if all agents play an SBR, then their joint policy is called a Logit equilibrium (LE), which is a subset of the more general Quantal Response equilibrium (McKelvey & Palfrey, 1995). In some literature, the Logit equilibrium is also called Stochastic equilibrium (Maher, 1998; Akamatsu, 1996). We compute the LE by Stochastic Fictitious Play (Hofbauer & Sandholm, 2002). In detail, each agent starts from a random policy and iteratively computes the SBR to the other policies. This process is repeated with a step size annealing according to Nagurney and Zhang (1996) until a fixed point is reached (see Appendix B.2 for details).

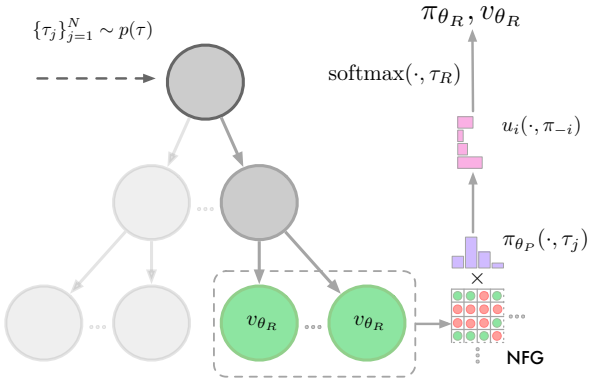
In the LE, both players play with the same rationality. But, it is also useful to model asymmetric rationality, i.e. one rational agent playing with one or multiple other imperfect agents. An equilibrium that models such asymmetry is the Quantal Stackelberg Equilibrium (QSE) (Milec et al., 2021), which is restricted to games of two agents. One rational agent maximizes their reward given that the other weak agent plays an SBR: $\pi_i = \arg\max u_i(\pi_i, SBR(\pi_i, \tau))$. However, the weak agent can only play an SBR if they know the optimal policy of the rational agent. This assumption is true in repeated interactions as one can observe the policy of the other agent, but violated in zero-shot interactions.

4. Method

In contrast to existing methods, we do not use an ensemble to train with diverse agents, but rather focus on training against agents of different playing strengths. This strength is parameterized by a scalar temperature τ. We consider the following setup: a perfectly rational agent i plays a game with one or multiple weak agents, who only inherit a bounded rationality. Therefore, each of the weak agents $j \in -i$ is modeled by a scalar τ_j.



(a) The **proxy** model evaluates the leaf nodes of its fixed-depth search tree using a learned value function v_{θ_P}. It approximates a Logit equilibrium (LE) given the temperature τ with its policy π_{θ_P}. The backup operator is a solver for the LE of a Normal-form game constructed from sibling nodes. The temperature τ is sampled uniformly transformed by a cosine (Appendix I.1).



(b) The **response** model evaluates the leaf nodes of its fixed-depth search tree using a learned value function v_{θ_R}. It computes the action utilities $u_i(\cdot, \pi_{-i})$ induced by the trained proxy policy π_{θ_P} given temperatures $\tau_1, \tau_2, \ldots, \tau_N$. The policy π_{θ_R} approximates a Smooth Best Response (SBR) with fixed response temperature τ_R to the action utilities.

Figure 3. Training architecture of the proxy and response models of Albatross. Both models are trained via planning-augmented self-play using fixed-depth search and are conditioned on one (proxy model) or multiple (response model) temperatures τ that are drawn from a distribution $p(\tau)$. The response model uses the trained proxy model to compute the Smooth Best Response Logit Equilibrium (SBRLE).

4.1. Smooth Best Response Logit Equilibrium (SBRLE)

Existing concepts of asymmetric equilibria, i.e. QNE and QSE, are a model for the interaction of a single weak agent and a perfectly rational agent. In detail, the weak agent plays an SBR to the respective optimal policy of the rational agent. However, this implies that the optimal policy of the rational agent is known to the weak agent. In other words, their bounded rationality only prevents them from computing their own optimal strategy, but not of the rational player. This assumption is valid in repeated interactions where the other agents policy can simply be observed, but violated in our setup of zero-shot interactions.

To model the symmetric bounded rationality in a setup with multiple weak agents, we utilize Logit equilibria (LE). The LE does not need to be the same for all bounded rational agents, i.e. they could play according to different LE with different temperatures. Given the weak agents play according LEs, we can compute a (smooth) best response to their policies. We call the joint policy of LE and the rational agents response a *Smooth Best Response Logit Equilibrium (SBRLE)*: $(SBR(\pi_{-i}, \tau_R), \pi_{-i})$ is a SBLRE, iff $\forall\, j \in -i$ π_j is a LE policy with temperature τ_j, where τ_R is the response temperature of the rational player. If $\tau_R = \infty$, then the rational player plays a best response to the weak agents and we call the corresponding joint policy a *Best Response Logit equilibrium (BRLE)*.

To highlight the difference between QSE and SBRLE: In the QSE, the weak agent plays an SBR to the optimal policy of the rational agent, but in the SBRLE to the LE policy, if the rational agent were to play according to LE. The different equilibria are visualized in Figure 2 for a zero-sum NFG. Even though the NE achieves maximal utility against perfectly rational opponents, it does not gain any utility from playing a weak opponent (in the example, NE lies on the intersection of BR and SBR). Depending on the NFG and rationality of agents, the quantal equilibria yield higher expected utility. In the SBRLE, the rational agent can freely choose the response temperature τ_R. A high response temperature directly corresponds to higher utility, but a response temperature of $\tau_R = \infty$ may impair the training process, since a BR is not necessarily unique. Consequently, the SBRLE is preferable to BRLE as it yields a unique learning target. For a detailed discussion on the effect of a unique learning target, see Appendix D.

4.2. Albatross

The computation of an SBRLE requires a full traversal of the game tree for every agent, initially to compute all Logit equilibria and afterwards the BR of the rational player. Additionally, a complete re-computation is required for every new temperature estimate of the weak agents. Since this is infeasible for most games, we present Albatross, which approximates an SBRLE using neural networks. The training of Albatross consists of two stages. Firstly, a proxy model approximates Logit equilibria at different temperatures. Then, a response model is trained to exploit the proxy model. Both models are trained using an adaptation of AlphaZero. We briefly outline the original AlphaZero algorithm, but refer to Appendix A and the original paper

(Silver et al., 2018; 2017) for a detailed explanation. Using AlphaZero, an actor-critic network predicting policy $\pi_\theta(o_i)$ and value $v_\theta(o_i)$ of agent i is trained, where o_i is an observation of the current game state from the perspective of agent i. During training, Monte-Carlo tree search (MCTS) (Metropolis & Ulam, 1949) is used as policy improvement operator, i.e. the policy of the root node is used as target for gradient updates. Targets for the gradient updates of the value function are the cumulative rewards of an episode, i.e. the result of a Monte-Carlo policy evaluation. During MCTS, the value function is used as an evaluation heuristic in the leaf nodes and the policy as guidance for exploration.

Similar to the original AlphaZero algorithm, the proxy model predicts policy $\pi_{\theta_P}(o_i, \tau)$ and value $v_{\theta_P}(o_i, \tau)$ of agent i based on observation o_i, but is also conditioned on the temperature τ. This conditioning allows the prediction of policy and value for LE of any temperature in the training distribution. During training, the temperature τ is sampled at the beginning of an episode from a distribution $p(\tau)$ in the interval $[\tau_{min}, \tau_{max}]$ (see Appendix I.1 for details regarding the training distribution). For the policy- and value improvement operator, we utilize fixed depth search instead of MCTS. In detail, we start at the leaf nodes and construct an NFG from sibling nodes. Then, we use a solver to compute the Logit equilibrium of the NFG and propagate the expected utility of the equilibrium to the parent node. This process is repeated until the LE at the root node is known. Then, the policy and value at the root node are used as targets for the gradient updates.

After the proxy model is trained, we start training of the response model. In contrast to the single scalar temperature τ of the proxy model, the response model is conditioned on the temperature of every agent except itself, which we define as $\tau_{-i} = (\tau_1, \ldots, \tau_{i-1}, \tau_{i+1}, \ldots, \tau_N)$. Therefore, the response model predicts policy $\pi_{\theta_R}(o_i, \tau_{-i})$ and value $v_{\theta_R}(o_i, \tau_{-i})$. Again, we utilize fixed depth search and an adapted backup function for training. During backup, we approximate the policies π_{-i} of other agents with the policy π_{θ_P} of the proxy model. Given the proxy policies, we compute the smooth best response $SBR(\pi_{-i,\theta_P}, \tau_R)$, where the response temperature τ_R is a fixed hyperparameter. The policy at the root node as well as its expected utility are used as targets for gradient updates. The policy π_{θ_R} of the response model is used for evaluation. We visualize the training scheme of proxy and response model in Figure 3. Detailed pseudocode for training AlphaZero and Albatross can be found in Appendix A.

4.3. Online Temperature Estimation

The previous two sections presented the methodology for training a policy conditioned on the temperature of other agents. At test-time, the rational agent has to estimate the temperature of the weak agents to input these temperatures into the response model. As a first option, the temperature can originate from insights about the agent. For example, a uniformly random agent always plays with temperature of $\tau = 0$. In zero-shot interactions, no previous knowledge about the other agent exists, but it is possible to estimate their temperature online using *Maximum Likelihood Estimation (MLE)*. We consider the following setup: a rational agent i intents to estimate the temperatures τ_j of weak agents $j \in -i$. Given K observations of actions $(a_j^1, \ldots, a_j^K)$ and corresponding optimal policies $(\pi_{-j}^1, \ldots, \pi_{-j}^K)$ of the other agents, one can estimate the temperature τ_j of weak agent j. The log-likelihood $l(\tau_j)$ of player j exhibiting temperature τ_j can be computed (Reverdy & Leonard, 2015) as

$$l(\tau_j) = \sum_{k=1}^{K} \left[\tau_j \, u_j^k(a_j^k, \pi_{-j}^k) - \ln \sum_{a_j \in A_j} \exp\left(\tau_j \, u_j^k(a_j, \pi_{-j}^k)\right) \right],$$

$$\frac{\partial l}{\partial \tau_j} = \sum_{k=1}^{K} \left[u_j^k(a_j^k, \pi_{-j}^k) - \frac{\sum_{a_j \in A_j} u_j^k(a_j, \pi_{-j}^k) \exp\left(\tau_j \, u_j^k(a_j, \pi_{-j}^k)\right)}{\sum_{a_j \in A_j} \exp\left(\tau_j \, u_j^k(a_j, \pi_{-j}^k)\right)} \right].$$

In order to find the maximum likelihood, one can utilize the gradient of the likelihood function $\frac{\partial l}{\partial \tau_j}$. The global optimum can be computed using simple line search over the temperature τ_j, because the likelihood function is concave, which we prove in Appendix F. Note that globally optimal policies are not well defined if multiple Nash equilibria exist due to the equilibrium selection problem (Punniyamoorthy et al., 2023). Therefore, we define optimal play in relation to the learned equilibrium of the proxy model. Specifically, we use the policy $\pi_{\theta_P}(\cdot, \tau_{max})$ of the proxy model with the highest temperature τ_{max}.

5. Empirical Evaluation

In our experiments, we want to research the following questions: (**Q1**) How does Albatross cooperate with unknown agents and adapt its behavior? (**Q2**) What effect has the temperature on its behavior? (**Q3**) Is Albatross able to estimate rationality within a single episode? (**Q4**) Can Albatross exploit weak enemies in the competitive domain? We answer these questions in the cooperative game of Overcooked (Carroll et al., 2019) and the competitive game of Battlesnake (Chung et al., 2020). In all experiments, we evaluate on five different seeds.

5.1. Cooperative Overcooked

For cooperative tasks, we evaluate Albatross in the Overcooked benchmark. In this game, two agents are placed in a kitchen and tasked with cooking as many soups as possible

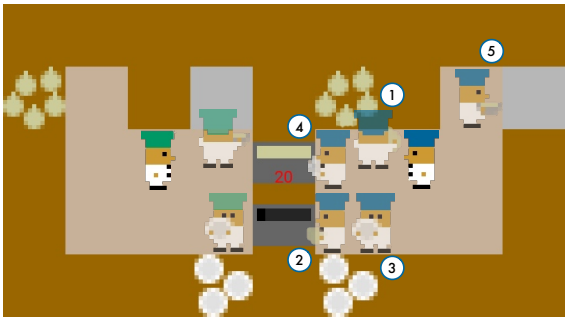


Figure 4. Albatross agent (right) and a possibly weak agent (left) in the Asymmetric Advantage layout of Overcooked. If the left agent plays rationally, they should realize that they have a shorter path to their serving location (gray tile). They would move down, retrieve a dish and deliver the soup. Having a strong estimation of rationality (e.g. $\tau = 10$), Albatross trusts them to deliver the soup and moves up to pick up an onion ① and prepare the next soup in the other pot ②. If Albatross has an estimation of weak rationality for the left agent (e.g. $\tau = 0$), then Albatross moves down to retrieve a dish ③, collects the soup ④ and serves it themselves ⑤.

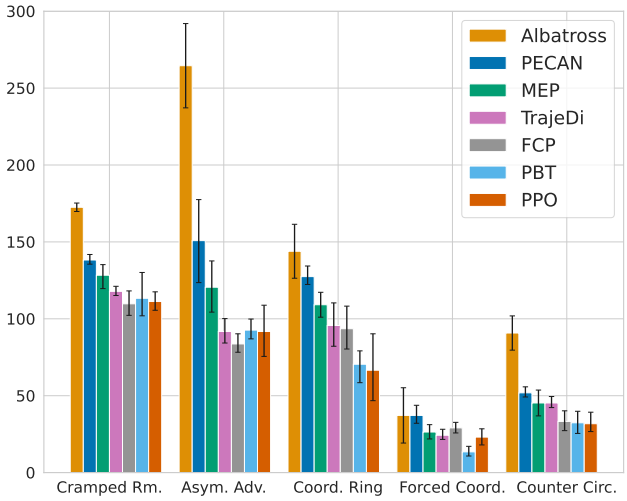


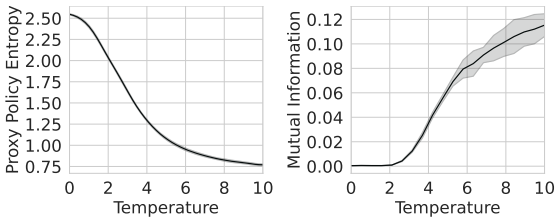
Figure 5. Cooperation performance with a behavior cloning agent trained on a dataset of human play (Carroll et al., 2019) in all five layouts of Overcooked. Episodes last 400 time steps and agents receive a common reward of 20 for delivering a soup.

in a given time frame. The agents may perform six actions: move up, down, left, right, stay in place or interact with the environment. To cook a soup, an agent firstly needs to fetch and place an onion in a pot three times. Then, they have to start the cooking process, wait for 20 steps and retrieve a dish from a dish dispenser. Lastly, they have to put the soup on the dish and serve the soup at a counter. There exist five different kitchen layouts (see Appendix G).

To answer (**Q1**) qualitatively, we show an example of the adaptive behavior of Albatross in Figure 4. Albatross performs different action sequences depending on the rationality estimation of the other agent. For a quantitative answer, we simulate the cooperation with humans by evaluating with a behavior cloning agent trained on human data. We compare Albatross to Proximal Policy-Optimization (Schulman et al., 2017), Population-based Training (Jaderberg et al., 2017), Fictitious Co-Play (Strouse et al., 2021), Trajectory Diversity (Lupu et al., 2021), Maximum Entropy Population-Based training (Zhao et al., 2023) and PECAN (Lou et al., 2023). In Figure 5, the reward in cooperation with the behavior cloning agent is displayed. In all layouts, except Forced Coordination, Albatross yields higher cooperation rewards than all baseline methods. On average, Albatross outperforms PECAN by 37.6%. In the Forced Coordination layout, good behavior is difficult to learn as rewards highly depend on the actions of the other agents.

Additionally, we evaluate the zero-shot cooperation performance of Albatross with different scripted agents, that exhibit specific behavioral pattern (Yu et al., 2023). For example, we pair the learned policy with agents that only place onions in a pot, or place dishes everywhere in the kitchen. Results of the evaluation are displayed in Table 1. The scripted policies are out-of-distribution regarding the training of Albatross, since they exhibit specific irrational behavior regardless of the reward. In contrast, Albatross only trained with Logit equilibrium policies, which exhibit an error probability proportional to the expected reward per action. Nevertheless, in most settings Albatross achieves cooperation performance greater or equal to Hidden-Utility Self-Play (HSP) (Yu et al., 2023), whose event-based training models such biased policies. This indicates that Albatross learns best responses which are robust against violations of the modelling assumptions.

To analyze the effect of temperature on the behavior of Albatross (**Q2**), we firstly analyze proxy and response model

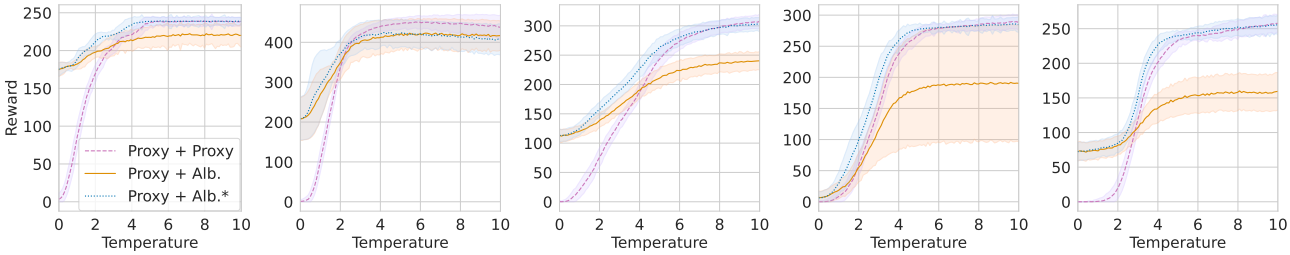


(a) Proxy Policy Entropy (b) MI Proxy & Response

Figure 6. Effects of temperature on the behavior of Albatross in the Counter Circuit layout. (a) The entropy of the proxy model decreases with rising temperature and (b) the mutual information between response and proxy policy increases.

Table 1. Cooperation performance with scripted agents exhibiting specific behaviors, e.g. always placing onions in the cooking pot. None of the behavior patterns were used during training, such that the learned policy has to adapt in zero-shot coordination.

ENVIRONMENT	SCRIPT	FCP	MEP	TRAJDIV	HSP	ALBATROSS
ASYM. ADV.	ONION PLACEMENT	334.8±13.0	330.5±14.2	323.6±17.0	**376.8±9.9**	342.5±11.3
ASYM. ADV.	ONION PLAC.+DELIVERY	297.7±3.4	298.5±3.4	290.0±4.7	300.1±4.1	**309.2±11.1**
COORD. RING	ONION EVERYWHERE	109.1±7.9	124.0±3.4	116.9±8.9	121.2±12.6	**143.9±2.4**
COORD. RING	DISH EVERYWHERE	94.4±3.8	100.2±5.3	107.3±5.3	115.4±7.4	**117.4±0.7**
COUNTER CIRC.	ONION EVERYWHERE	63.7±9.2	88.9±5.1	82.0±12.8	107.5±3.5	**119.6±1.2**
COUNTER CIRC.	DISH EVERYWHERE	57.0±5.3	53.0±1.8	57.2±2.2	**78.5±4.1**	**78.9±4.3**



(a) Cramped Room (b) Asym. Advantage (c) Coord. Ring (d) Forced Coord. (e) Counter Circuit

Figure 7. Self-play performance of the proxy model as well as cooperation performance between Albatross and the proxy model at different temperatures. Albatross* denotes the cooperation capability of Albatross without Maximum Likelihood Estimation, i.e. if the the true fixed temperature of the proxy model is given.

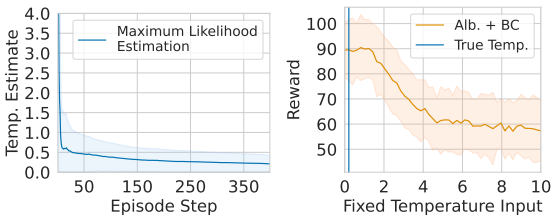
directly. In Figure 6, we show that the entropy of the proxy policy decreases with rising temperature. This corresponds to the lower error probability of agents with higher rationality. To further highlight the adaptive behavior of Albatross, we measure the mutual information (Li et al., 2022; 2023) between its response- and proxy policy:

$$I(\pi_{\theta_R}; \pi_{\theta_P}) = H(\pi_{\theta_P}) - H(\pi_{\theta_P} \mid \pi_{\theta_R}).$$

With rising temperatures, we observe a drop in the entropy of the proxy policy with an increase in mutual information, estimated via the conditional action frequencies. This captures the level of cooperation between both policies as it implies a decrease in the conditional entropy $H(\pi_{\theta_P} \mid \pi_{\theta_R})$. Consequently, Albatross cooperates with rational agents and acts self-reliant if the other agent does not cooperate.

Next, we analyze the effect of temperature on the expected reward in Figure 7. For temperature $\tau = 0$, the proxy model learned a uniformly random policy and does not achieve any reward in self-play. At higher temperatures, the achieved reward directly corresponds to the temperature. Analyzing Albatross with the proxy model shows the cooperation with agents of different playing strength. For example, the reward attainable when cooperating with a uniformly random agent is visible at $\tau = 0$. Given an exact temperature estimation (denoted as Albatross*), we expect the reward of Albatross* with the proxy model to converge to the proxy self-play performance at high temperatures as both play optimally. We can observe this effect in all layouts except Asymmetric Advantage, where the training of the response model did not perfectly converge.

To show that Albatross is able to estimate rationality within a single episode (**Q3**), we can observe the difference in reward between Albatross and Albatross* in Figure 7. At high temperatures, the reward obtained by Albatross is lower than the self-play performance of the proxy model due to the aleatoric uncertainty of the MLE. The extend of this uncertainty depends on the layout. In Figure 8, we show the result of MLE at different time steps during an episode in cooperation with the human behavior cloning agent. After



(a) MLE during Episode (b) Temp. Misspecification

Figure 8. Analysis of temperature estimation of Albatross in the Counter Circuit layout. (a) During an episode, temperature estimation of MLE quickly converges to the true value. (b) We analyze the the expected reward using a fixed temperature input.

(a) Tron (b) Stochastic 2 Player (c) Stochastic 4 Player

Figure 9. Evaluation of AlphaZero and Albatross against baseline MCTS (+ value function heuristic) agents with different budgets of search iterations in the game modes of Battlesnake. Albatross is able to exploit weak agents better than AlphaZero.

only few time steps, MLE converges towards the true temperature estimate. An evaluation of Albatross with a fixed temperature input reveals that minor estimation errors have little effect on the achieved reward. However, major overestimation of the other agents rationality leads to a significant drop in performance.

In Figure 8, after only few episode steps in an episode, the MLE already estimates a temperature close to the true value. An evaluation of Albatross with a fixed temperature input reveals that minor estimation errors have little effect on the achieved reward. However, major overestimation of the other agents rationality leads to a significant drop in performance.

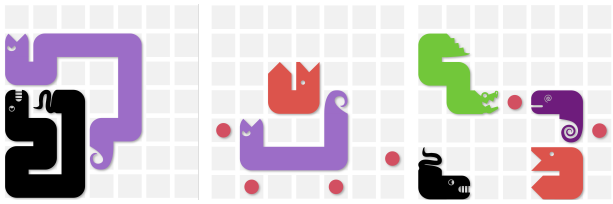
5.2. Competitive Battlesnake

To show that Albatross is able to exploit weak agents in the competitive domain (**Q4**), we evaluate in the game of Battlesnake. The game takes place on a grid, where agents have to survive as long as possible. They die, if they collide either with a wall or the body of a snake. If two agents collide head-to-head, the longer snake survives. In the stochastic extensions, food spawns randomly on the map which agents have to eat to prevent starving and grow their body. In contrast, there exist no food in the game of Tron and the body of each snake is elongated in every turn. All three game modes are visualized in Figure 10. Agents receive a reward of $+1$ for winning and -1 for dying in the game modes with two agents. In the mode with four agents, a reward of $+1$ is distributed among the living agents if another agent dies.

We compare Albatross against AlphaZero (Silver et al., 2018). Since AlphaZero was developed for sequential games, we perform an extensive analysis on the adaptation of AlphaZero to simultaneous games (see Appendix C). For a fair comparison, Albatross and AlphaZero are trained on the same time budget and hardware. Since training Albatross is a two-step process, proxy and response model are trained for half as long as AlphaZero.

In Figure 9, both play against baseline agents, which utilize simultaneous-move MCTS (Bošanský et al., 2016) with a handcrafted value heuristic adapted from Schier and Wüstenbecker (2019). Rationality of the baseline agents is modulated by the compute budget, i.e. the number of tree search iterations. Albatross consistently outperforms AlphaZero and the reward difference is highest against weaker enemies. That is, because they violate the perfect rationality assumption of AlphaZero. In contrast, Albatross is able to identify and exploit their weak rationality. Strong enemies are not exploitable, which leads to a convergence of the reward achieved by Albatross and AlphaZero. Interestingly, in the stochastic 2-player mode, the reward difference between Albatross and AlphaZero is highest for medium rational agents with about 10^3 iterations. Those agents are able to surprise AlphaZero by playing suboptimal, but still good enough to win some games. This effect does not occur in the game of Tron as games are shorter and mistakes result in a quick death.

Additionally, we play a tournament between Albatross, AlphaZero and baseline agents of different strengths in the stochastic mode with four agents. In Figure 1, the results of this tournament are displayed. We plot the estimated temperature of the agents and the TrueSkill scores achieved in the tournament. For the rationality of Albatross we use the



(a) Tron (b) Stoch. 2 Player (c) Stoch. 4 Player

Figure 10. Game modes of Battlesnake: the standard game of Tron as well as the stochastic extensions of Battlesnake for either two or four agents. Food is visualized by red circles.

response temperature $\tau_R = 10$. AlphaZero achieves nearly maximum temperature of $\tau = 9.99$, which corresponds to optimal play assuming all other agents play optimal as well. However, the baseline agents do not play optimally, which Albatross is able to detect and exploit. Consequently, it achieves a higher TrueSkill score than AlphaZero.

5.3. Cooperative Battlesnake

Lastly, we evaluate the capabilities of Albatross to cooperate with more than two agents. To this end, we devise a game variant of Tron with cooperative rewards, i.e. the objective of all agents is to stay alive as a group for as long as possible. Since the previously used board size of 7×7 is too small for four players in the deterministic game mode of Tron, we increase the board size to 11×11. The results are shown in Figure 11. Due to the deterministic game dynamics, the board fills up quickly and games are short. Therefore, the possible variation in discounted reward is also small. However, Albatross still outperforms AlphaZero by a small margin when playing with weak partners, i.e. few search iterations. Again, this difference diminishes with partners of higher rationality since AlphaZero assumes optimal play. These results verify that Albatross is able to cooperate well with more than two agents of different rationality.

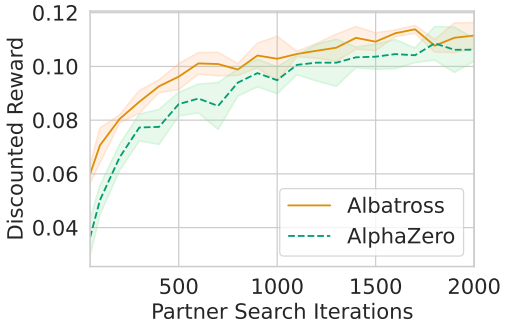


Figure 11. Evaluation of AlphaZero and Albatross n a four player cooperative game variant of Tron with baseline MCTS (+ value function heuristic) agents that have different budgets of search iterations i.

6. Limitations and Future Work

To accurately estimate the rationality of an agent, Albatross requires observations of their behavior. In Figure 8, we demonstrate that this estimation quickly converges within 20 to 30 time steps. This is exemplified in the game of Tron, which has a maximum game length of 24. However, Albatross may not be applicable to games with even shorter episodes. In future work, Albatross could incorporate a prior temperature likelihood and perform maximum a posteriori estimation to be applicable to very short interactions. For example, a prior likelihood could be obtained from an online leaderboard or other sparse knowledge about the other agent. Additionally, Unsupervised Reinforcement Learning could be used to obtain a prior policy (Schubert et al., 2023). Another current limitation of Albatross is the dependency on planning in the joint action space of all agents. The size of the joint action space grows exponentially with the number of agents and number of actions per agent. Therefore, the tree search becomes a weak improvement operator if only a fraction of nodes can be evaluated. This prevents the application in domains with large joint action spaces or domains where the environment dynamics are unknown. For example, Albatross is not applicable to the MeltingPot environment (Agapiou et al., 2022) as it does not allow for planning. In future work, Albatross can be enhanced with a learned environment model akin to MuZero (Schrittwieser et al., 2020) to address these limitations.

7. Conclusion

We developed the novel equilibrium concept of an SBRLE for modeling the interaction between a single rational and multiple weak agents in zero-shot interactions. Since the SBRLE is infeasible in most games, we proposed Albatross, which is capable of learning the SBRLE through a combination of self-play and planning. Using Albatross, we are able to reach state of the art in the Overcooked benchmark for cooperative tasks. We showed that Albatross is able to estimate rationality of unknown agents within a single episode and cooperate with them by adapting its behavior. Analyzing the effect of temperature estimation, we find Albatross cooperates with rational partners and behaves self-reliant with weak partners. Moreover, we showed that Albatross is able to exploit weak enemies in the competitive domain.

Impact Statement

This paper represents work contributing to the advancement of Human-AI collaboration. A potential societal consequence of our work is the adoption of adaptive AI-Assistants, increasing quality of life on the human side. However, we are aware that rationality is an intricate concept, which cannot completely be modeled using a scalar value. For real-world applications, aspects of fairness, privacy and safety must be examined. Moreover, for a given application, ethical aspects must be considered to determine if estimating the rationality of a human is appropriate.

Acknowledgements

This work was supported by the Federal Ministry of Education and Research (BMBF), Germany under the AI service center KISSKI (grant no. 01IS22093C), the Lower Saxony Ministry of Science and Culture (MWK) through the

zukunft.niedersachsen program of the Volkswagen Foundation and the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122) and (RO 2497/17-1).

References

Agapiou, J. P., Vezhnevets, A. S., Duéñez-Guzmán, E. A., Matyas, J., Mao, Y., Sunehag, P., Koster, R., Madhushani, U., Kopparapu, K., Comanescu, R., Strouse, D., Johanson, M. B., Singh, S., Haas, J., Mordatch, I., Mobbs, D., and Leibo, J. Z. Melting Pot 2.0. *ArXiv*, abs/2211.13746, 2022.

Akamatsu, T. Cyclic flows, Markov process and stochastic traffic assignment. *Transportation Research Part B: Methodological*, 30(5):369–386, October 1996.

Anthony, T., Tian, Z., and Barber, D. Thinking Fast and Slow with Deep Learning and Tree Search. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. The Nonstochastic Multiarmed Bandit Problem. *SIAM J. Comput.*, 32:48–77, 2002.

Bošanský, B., Lisý, V., Lanctot, M., Čermák, J., and Winands, M. H. Algorithms for computing strategies in two-player simultaneous move games. *Artificial Intelligence*, 237:1–40, 2016.

Brown, N., Lerer, A., Gross, S., and Sandholm, T. Deep Counterfactual Regret Minimization. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 793–802. PMLR, 09–15 Jun 2019.

Carroll, M., Shah, R., Ho, M. K., Griffiths, T. L., Seshia, S. A., Abbeel, P., and Dragan, A. On the Utility of Learning about Humans for Human-AI Coordination. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 5174–5185, Red Hook, NY, USA, 2019. Curran Associates Inc.

Cerny, J., Lisý, V., Bošanský, B., and An, B. Dinkelbach-Type Algorithm for Computing Quantal Stackelberg Equilibrium. In Bessiere, C. (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 246–253. International Joint Conferences on Artificial Intelligence Organization, 7 2020.

Cesa-Bianchi, N., Gentile, C., Neu, G., and Lugosi, G. Boltzmann Exploration Done Right. In *NIPS*, 2017.

Chung, J., Luo, A., Raffin, X., and Perry, S. Battlesnake Challenge: A Multi-agent Reinforcement Learning Playground with Human-in-the-loop. *ArXiv*, abs/2007.10504, 2020.

Dinkelbach, W. On Nonlinear Fractional Programming. *Management Science*, 13(7):492–498, 1967.

Dockhorn, A. and Kruse, R. Combining cooperative and adversarial coevolution in the context of Pac-Man. In *2017 IEEE Conference on Computational Intelligence and Games, CIG 2017*, pp. 60–67, 2017. ISBN 9781538632338. doi: 10.1109/CIG.2017.8080416.

Dockhorn, A., Doell, C., Hewelt, M., and Kruse, R. A decision heuristic for Monte Carlo tree search doppelkopf agents. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, November 2017. ISBN 978-1-5386-2726-6. doi: 10.1109/SSCI.2017.8285181.

Everett, R. and Roberts, S. J. Learning Against Non-Stationary Agents with Opponent Modelling and Deep Reinforcement Learning. In *AAAI Spring Symposia*, 2018.

Fang, F., Nguyen, T. H., Pickles, R., Lam, W. Y., Clements, G. R., An, B., Singh, A., Schwedock, B. C., Tambe, M., and Lemieux, A. PAWS — A Deployed Game-Theoretic Application to Combat Poaching. *AI Mag.*, 38(1):23–36, mar 2017.

Fudenberg, D. and Levine, D. *The Theory of Learning in Games*. Economics Learning and Social Evolution Series. MIT Press, 1998.

Herbrich, R., Minka, T., and Graepel, T. TrueSkill™: A Bayesian Skill Rating System. In Schölkopf, B., Platt, J., and Hoffman, T. (eds.), *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

Hofbauer, J. and Sandholm, W. H. On the Global Convergence of Stochastic Fictitious Play. *Econometrica*, 70(6): 2265–2294, 2002.

Howard, A. G., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., and Adam, H. Searching for mobilenetv3. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019.

Hu, H., Lerer, A., Peysakhovich, A., and Foerster, J. "Other-Play" for Zero-Shot Coordination. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4399–4410. PMLR, 13–18 Jul 2020.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., and Kavukcuoglu, K. Population Based Training of Neural Networks. *ArXiv*, abs/1711.09846, 2017.

Jeon, M., Lee, J., and Ko, S.-K. Modular reinforcement learning for playing the game of tron. *IEEE Access*, 10: 63394–63402, 2022.

Knegt, S. J. L., M. Drugan, M., and A. Wiering, M. Opponent Modelling in the Game of Tron using Reinforcement Learning:. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence*, pp. 29–40, Funchal, Madeira, Portugal, 2018. SCITEPRESS - Science and Technology Publications.

Lan, L.-C., Zhang, H., Wu, T.-R., Tsai, M.-Y., Wu, I.-C., and Hsieh, C.-J. Are AlphaZero-like Agents Robust to Adversarial Perturbations? In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 11229–11240. Curran Associates, Inc., 2022.

Lanctot, M., Wittlinger, C., Den Teuling, N., and Winands, M. Monte Carlo tree search for simultaneous move games: A case study in the game of Tron. *BNAIC 2013*, pp. 104–111, 01 2013.

Leyton-Brown, K. and Shoham, Y. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Springer International Publishing, Cham, 2008.

Li, P., Tang, H., Yang, T., Hao, X., Sang, T., Zheng, Y., Hao, J., Taylor, M. E., and Wang, Z. PMIC: Improving Multi-Agent Reinforcement Learning with Progressive Mutual Information Collaboration. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12979–12997. PMLR, 17–23 Jul 2022.

Li, P., Hao, J., Tang, H., Zheng, Y., and Fu, X. RACE: Improve multi-agent reinforcement learning with representation asymmetry and collaborative evolution. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 19490–19503. PMLR, 2023.

Lin, J. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

Lisy, V., Kovarik, V., Lanctot, M., and Bosansky, B. Convergence of Monte Carlo Tree Search in Simultaneous Move Games. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Liu, H. X., He, X., and He, B. Method of Successive Weighted Averages (MSWA) and Self-Regulated Averaging Schemes for Solving Stochastic User Equilibrium Problem. *Networks and Spatial Economics*, 9:485–503, 2009. doi: 10.1007/s11067-007-9023-x.

Loshchilov, I. and Hutter, F. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2017.

Lou, X., Guo, J., Zhang, J., Wang, J., Huang, K., and Du, Y. PECAN: Leveraging Policy Ensemble for Context-Aware Zero-Shot Human-AI Coordination. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, pp. 679–688, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450394321.

Lupu, A., Cui, B., Hu, H., and Foerster, J. Trajectory Diversity for Zero-Shot Coordination. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7204–7213. PMLR, 18–24 Jul 2021.

Maher, M. Algorithms for logit-based stochastic user equilibrium assignment. *Transportation Research Part B: Methodological*, 32(8):539–549, 1998. ISSN 0191-2615. doi: https://doi.org/10.1016/S0191-2615(98)00015-0.

McFadden, D. Conditional Logit Analysis of Qualitative Choice Behaviour. In Zarembka, P. (ed.), *Frontiers in Econometrics*, pp. 105–142. Academic Press New York, New York, NY, USA, 1973.

McKelvey, R. D. and Palfrey, T. R. Quantal Response Equilibria for Normal Form Games. *Games and Economic Behavior*, 10(1):6–38, 1995.

Metropolis, N. and Ulam, S. The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247): 335–341, September 1949.

Milec, D., Černý, J., Lisý, V., and An, B. Complexity and Algorithms for Exploiting Quantal Opponents in Large Two-Player Games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 5575–5583, May 2021.

Nagurney, A. and Zhang, D. *Projected Dynamical Systems and Variational Inequalities with Applications*, volume 2. Springer New York, 1996. doi: https://doi.org/10.1007/978-1-4615-2301-7.

Nash, J. Non-Cooperative Games. *Annals of Mathematics*, 54(2):286–295, 1951.

Nashed, S. B. and Zilberstein, S. A Survey of Opponent Modeling in Adversarial Domains. *J. Artif. Intell. Res.*, 73:277–327, 2022.

Papoudakis, G. and Albrecht, S. V. Variational Autoencoders for Opponent Modeling in Multi-Agent Systems. *CoRR*, abs/2001.10829, 2020.

Pavlidis, T. *Algorithms for Graphics and Image Processing*. Springer, 1982.

Polyak, B. T. New method of stochastic approximation type. *Autom. Remote Control*, 51:937–946, 1990.

Porter, R., Nudelman, E., and Shoham, Y. Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior*, 63(2):642–662, July 2008.

Punniyamoorthy, M., Abraham, S., and Thoppan, J. J. A Method to Select Best Among Multi-Nash Equilibria. *Studies in Microeconomics*, 11(1):101–127, 2023. doi: 10.1177/23210222211024388.

Reverdy, P. B. and Leonard, N. E. Parameter Estimation in Softmax Decision-Making Models With Linear Objective Functions. *IEEE Transactions on Automation Science and Engineering*, 13:54–67, 2015.

Robbins, H. and Monro, S. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3): 400–407, 1951. ISSN 00034851.

Rust, J. *Dynamic Programming*, pp. 3133–3158. Palgrave Macmillan UK, London, 2018.

Samothrakis, S., Robles, D., and Lucas, S. M. A UCT agent for Tron: Initial investigations. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pp. 365–371, August 2010.

Saverino, B. A Monte-Carlo Tree Search for playing Tron. Master's thesis, Montefiore, Department of Electrical Engineering and Computer Science, 2011.

Schier, M. B. and Wüstenbecker, N. Adversarial N-player Search using Locality for the Game of Battlesnake. In *SKILL 2019 - Studierendenkonferenz Informatik*, pp. 109–120. Gesellschaft für Informatik e.V., Bonn, 2019.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, December 2020.

Schubert, F., Benjamins, C., Döhler, S., Rosenhahn, B., and Lindauer, M. POLTER: Policy Trajectory Ensemble Regularization for Unsupervised Reinforcement Learning. *Transactions on Machine Learning Research*, April 2023. ISSN 2835-8856.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

Sebbane, M. Board Games from Canaan in the Early and Intermediate Bronze Ages and the Origin of the Egyptian Senet Game. *Tel Aviv*, 28(2):213–230, 2001. doi: 10.1179/tav.2001.2001.2.213.

Shafiei, M., Sturtevant, N. R., and Schaeffer, J. Comparing UCT versus CFR in Simultaneous Games, 2009.

Shapley, L. S. *1. Some Topics in Two-Person Games*, pp. 1–28. Princeton University Press, Princeton, 1964. ISBN 9781400882014. doi: doi:10.1515/9781400882014-002.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362 (6419):1140–1144, 2018.

Strouse, D., McKee, K., Botvinick, M., Hughes, E., and Everett, R. Collaborating with humans without human data. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 14502–14515. Curran Associates, Inc., 2021.

Tak, M. J. W., Lanctot, M., and Winands, M. H. M. Monte Carlo Tree Search variants for simultaneous move games. In *2014 IEEE Conference on Computational Intelligence and Games*, pp. 1–8, 2014.

Yang, R., Ordóñez, F., and Tambe, M. Computing optimal strategy against quantal response in security games. In *Adaptive Agents and Multi-Agent Systems*, 2012.

Yu, C., Gao, J., Liu, W., Xu, B., Tang, H., Yang, J., Wang, Y., and Wu, Y. Learning zero-shot cooperation with humans, assuming humans are biased. In *The Eleventh*

International Conference on Learning Representations, 2023.

Yu, X., Jiang, J., Zhang, W., Jiang, H., and Lu, Z. Model-Based Opponent Modeling. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Zhao, R., Song, J., Yuan, Y., Hu, H., Gao, Y., Wu, Y., Sun, Z., and Yang, W. Maximum Entropy Population-Based Training for Zero-Shot Human-AI Coordination. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37 (5):6145–6153, Jun. 2023.

A. Training Procedures of AlphaZero and Albatross

In this section, we outline the training procedure of Albatross and highlight the differences to the original AlphaZero algorithm. As an intuition, the proxy model learns to play like agents of different rationality, but only through self-play without external supervision (e.g. imitation learning). Afterwards, the response model learns to play optimally against the proxy model at all rationalities. In contrast, AlphaZero only performs self-play, only learning to play optimally against the own highly rational policy.

Therefore, the training of AlphaZero is very similar to the training of the proxy model, but the neural networks are not conditioned on the temperature. The main differences in the algorithms are highlighted in bold.

A.1. AlphaZero

In the original AlphaZero algorithm (Silver et al., 2018), Monte-Carlo tree search (MCTS) is used as a policy improvement operator. For sequential games this is a good choice as it allows for deep searches by only focusing on promising regions in the search space. However, in simultaneous games, the best action is highly dependent on the actions of the other players, which nullifies this advantage. This is reflected in our hyperparameter search (see Appendix C), where we found that replacing the standard MCTS with fixed depth search and Logit equilibrium backup yields the best performance.

Algorithm 1 Training of AlphaZero for simultaneous games

Input: Observation space $o \in \mathcal{O}$, maximum rationality τ_{max}
Output: Trained models $\pi_\theta(o_i)$ and $v_\theta(o_i)$
Initialize models π_θ and v_θ with random weights
Initialize replay buffer B
for each training episode **do**
  for each step of the episode **do**
    Perform fixed depth search to construct NFGs and compute LEs:
      1. At each leaf node, evaluate states via $v_\theta(o_{\text{leaf},i})$ and construct NFG from sibling nodes.
      2. Use a solver to compute the Logit equilibrium with temperature τ_{max} for the NFG
      3. Propagate the expected utility of the LE to the parent node
      4. Repeat until the LE at the root node is computed
    Let o_i be the state observation and $\pi_{\text{LE}}(o_i)$, $v_{\text{LE}}(o_i)$ be policy and value of the LE at the root node for each player i
    Collect experiences $(o_i, \pi_{\text{LE}}(o_i), v_{\text{LE}}(o_i))$ at this step and add to B
    Perform environment step by sampling actions a_i from $\pi_{\text{LE}}(o_i)$ for each player i
  end for
  for each minibatch sampled from B **do**
    Update π_θ and v_θ via CrossEntropy and MSE respectively
  end for
end for

The detailed training procedure is given in Algorithm 1. During each training step, a complete search tree is constructed up to a fixed depth. Then, the leaf nodes are evaluated by the critic network v_θ, which represent the utilities of Normal-form games (NFG) constructed from the direct sibling nodes. The Logit equilibrium of these NFGs is computed using an equilibrium solver (for details see Appendix B). The equilibria yield expected utilities for each player in the parent nodes, which again are used to create NFGs from sibling nodes. This process is repeated until the root node is reached. The Logit equilibrium at the root node yields an improved policy and value estimate for each player i at the current game state compared to the original policy and value prediction at this game state. These target values are added to the replay buffer and used later for gradient updates.

A.2. Albatross

The training of Albatross is a two-stage process. Firstly, a proxy model learns to imitate the behavior of agents at different rationality. Afterwards, a response model learns to play a smooth best response to the proxy model. The training process of the proxy model is very similar to the training of AlphaZero outlined above. This procedure is visualized in Algorithm 2. The major distinction is that policy and value networks are conditioned on a scalar temperature parameter, which controls

the rationality of the proxy model. To train the proxy model on a range of different temperatures, a temperature is sampled at the beginning of each training episode. During each step in the episode, again Logit equilibria are computed during fixed depth search for policy and value improvement. In contrast to the training of AlphaZero, the temperature sampled at the beginning of the episode is used for computing the Logit equilibria. Additionally, the temperature is also added to the replay buffer as they are necessary for the gradient updates.

Algorithm 2 Training of Proxy Model in Albatross (Differences to AlphaZero are highlighted in blue)

Input: Observation space $o \in \mathcal{O}$, temperature range $[\tau_{min}, \tau_{max}$, temperature distribution $p(\tau)$
Output: Trained proxy models $\pi_{\theta_P}(o_i, \tau)$ and $v_{\theta_P}(o_i, \tau)$
Initialize proxy models π_{θ_P} and v_{θ_P} with random weights
Initialize replay buffer B
for each training episode **do**
  Sample temperature τ from $p(\tau)$ within $[\tau_{min}, \tau_{max}]$
  for each step of the episode **do**
    Perform fixed depth search to construct NFGs and compute LEs:
      1. At each leaf node, evaluate states via $v_{\theta_P}(o_{\text{leaf},i}, \tau)$ and construct NFG from sibling nodes.
      2. Use a solver to compute the Logit equilibrium with temperature τ for the NFG
      3. Propagate the expected utility of the LE to the parent node
      4. Repeat until the LE at the root node is computed
    Let o_i be the state observation and $\pi_{\text{LE}}(o_i, \tau)$, $v_{\text{LE}}(o_i, \tau)$ be policy and value of the LE at the root node for each player i
    Collect experiences $(o_i, \pi_{\text{LE}}(o_i, \tau), v_{\text{LE}}(o_i, \tau)$, temperature $\tau)$ at this step and add to B
    Perform environment step by sampling actions a_i from $\pi_{\text{LE}}(o_i, \tau)$ for each player i
  end for
  for each minibatch sampled from B **do**
    Update π_{θ_P} and v_{θ_P} via CrossEntropy and MSE respectively
  end for
end for

After training the proxy model, the response model is trained using the policy of the proxy model. The policy and value network of the response model are not only conditioned on a single temperature, but a temperature for every player except itself. Therefore, in a game with n players, the response model is conditioned on $n-1$ scalar temperatures. The response model approximates the Smooth Best Response (SBR) to the policy of the proxy model. Since the response model should be trained on all combination of rationalities for the other agent, $n-1$ temperatures are sampled at the beginning of an episode. At each training step, a policy and value improvement is achieved by computing the SBRLE with tree search up to a fixed depth. At first, the leaf nodes of the search tree are evaluated using the value network v_{θ_R}, which form NFGs. The policy of the weak players in the NFG can be computed by using the proxy policy π_{θ_P} in the parent node with the respective temperature sampled at the beginning of the episode. Then, the SBR is computed by a softmax transformation with fixed response temperature τ_R on the expected utilities for player i. Similar to the training of AlphaZero and the Proxy model, the SBRLE is propagated to the root node and used for gradient updates. In contrast to the training of the proxy model, all $n-1$ temperatures are added to the replay buffer. Another difference to AlphaZero and the proxy model is the policy used for advancing the environment state between training steps. Only the action for the agent controlled by the response model is sampled from the SBRLE computed at the root node. All other actions are sampled from the proxy policy to accurately represent the state distribution when playing weak agents.

B. Algorithms for equilibrium computation

There exist a number of algorithms for computing the equilibria presented. We give a brief overview of the algorithms used in this work. All algorithms are implemented in C++ and available open source along with our code. To the best of our knowledge, this is the first open-source implementation of solvers for Quantal Stackelberg equilibria.

Algorithm 3 Training of Response Model in Albatross (Differences to Proxy are highlighted in blue)

Input: Observation space $o \in \mathcal{O}$, temperature range $[\tau_{min}, \tau_{max}]$, temperature distribution $p(\tau)$, fixed response temperature τ_R, proxy policy $\pi_{\theta_P}(o_i, \tau)$
Output: Trained response models $\pi_{\theta_R}(o_i, \tau_{-i})$ and $v_{\theta_R}(o_i, \tau_{-i})$
Initialize response models π_{θ_R} and v_{θ_R} with random weights
Initialize replay buffer B
for each training episode **do**
  Sample temperatures τ_{-i} from $p(\tau)$ within $[\tau_{min}, \tau_{max}]$ for each player except i
  for each step of the episode **do**
    Perform fixed depth search to approximate an SBRLE:
      1. At each leaf node, evaluate states via $v_{\theta_R}(o_{\text{leaf},i}, \tau_{-i})$ and construct NFG from sibling nodes.
      2. At the parent node evaluate the proxy policy for every other agent j as $\pi_{\theta_P}(o_{\text{parent},j}, \tau_j)$
      3. Compute the Smooth Best Response (SBR) to LE policies with response temperature τ_R
      4. Repeat until the SBR to the LE of the other player at the root node is computed
    Let o_i be the state observation and $\pi_{\text{SBRLE}}(o_i, \tau_{-i})$, $v_{\text{SBRLE}}(o_i, \tau_{-i})$ be policy and value of the SBRLE at the root node for player i
    Collect experiences $(o_i, \pi_{\text{SBRLE}}(o_i, \tau_{-i}), v_{\text{SBRLE}}(o_i, \tau_{-i})$, temperatures $\tau_{-i})$ at this step and add to B
    Perform environment step by sampling actions a_i from $\pi_{\text{SBRLE}}(o_i, \tau)$ for player i and from the proxy policy $\pi_{\theta_P}(o_j, \tau_j)$ for every other agent j
  end for
  for each minibatch sampled from B **do**
    Update π_{θ_R} and v_{θ_R} via CrossEntropy and MSE respectively
  end for
end for

B.1. Nash Equilibrium

For the computation of Nash equilibria, we use the algorithm of Porter et al. (2008), which is based on support enumeration. A support is the set of actions receiving a non-zero probability in the Nash equilibrium. Until an equilibrium is found, the supports are iterated and a linear program is solved to determine if a Nash equilibrium exists for the current support. Supports are ordered based on a heuristic prioritizing small and balanced supports. For games of more than two players, a non-linear program is solved. For details regarding the formulation of the (non-) linear program, we refer to the original work.

B.2. Logit Equilibrium

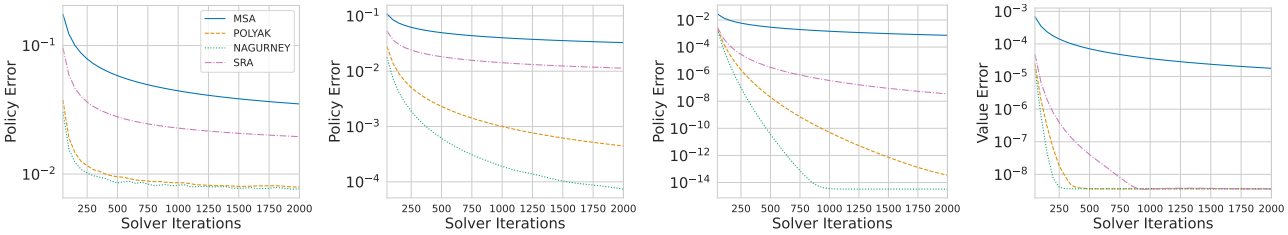
A Logit equilibrium can be computed by using the smooth best response dynamics. In detail, starting from a uniform policy, all players compute the smooth best response given the other players policy. These new policies are the basis for the computation of smooth best responses in the next iteration. This process is called *Stochastic Fictitious Play (SFP)* (Hofbauer & Sandholm, 2002) (sometimes also titled Smooth Fictitious Play (Fudenberg & Levine, 1998)). Simply following the dynamics yields a fixed point in two-player zero-sum games, but can form a cycle in other games (Shapley, 1964). Therefore, one has to anneal the step size for updating the policy of each player. Formally, in iteration t the policies are updated as $\pi_i^{t+1} = \pi_i^t + \alpha_t(SBR(\pi_{-i}^t, \tau) - \pi_i^t)$. Robbins and Monro (1951) proved that SFP converges almost surely to the equilibrium point for the step sizes $(\alpha_1, \alpha_2, \ldots)$ if the conditions $\lim_{t\to\infty} \alpha_t = 0$ and $\sum_{t=1}^{\infty} \alpha_t = \infty$ hold. We compare multiple step size schedules, which all fulfill the two mentioned conditions.

- The method of successive averages (MSA) (Robbins & Monro, 1951) updates the policy as an average of all previous policies, which is equivalent to using a step size of $\alpha_t = 1/t$.
- Polyak (1990) proposed to use a step size of $\alpha_t = t^{-2/3}$.
- Nagurney and Zhang (1996) proposed a schedule of learning rates which converges to zero at a slow rate: $(1, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots, t \text{ times } \frac{1}{t})$.
- Self-regulating average (SRA) (Liu et al., 2009) is a case-based schedule accelerating the decay if learning diverges

and decelerating the decay when converging. The learning rate factors are $\alpha_t = \frac{1}{\beta_t}$ with

$$\beta_t = \begin{cases} \beta_{t-1} + \gamma & \left\|\pi_i^t - SBR(\pi_{-i}^t, \tau)\right\| \geq \left\|\pi_i^{t-1} - SBR(\pi_{-i}^{t-1}, \tau)\right\| \\ \beta_{t-1} + \Gamma & \textit{else} \end{cases},$$

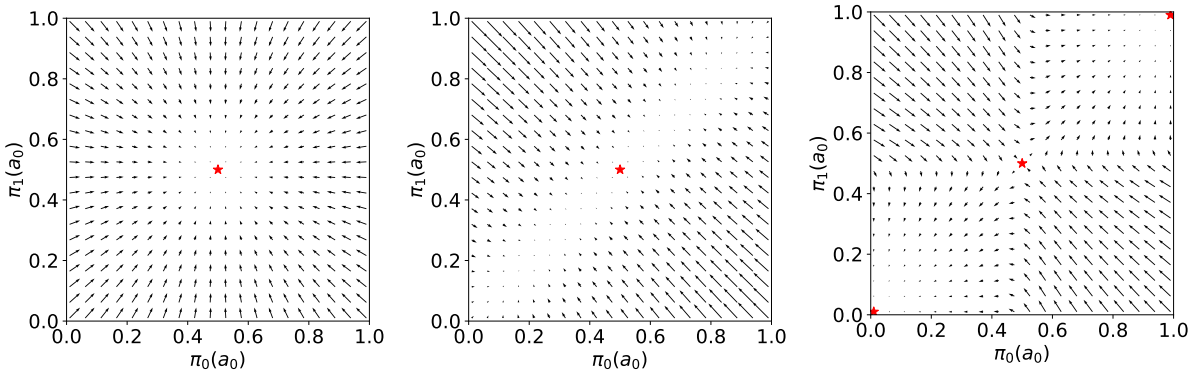
where $\gamma < 1$ and $\Gamma > 1$. For our experiments, we adopted the hyperparameters $\gamma = 0.3$ and $\Gamma = 1.8$ from Liu et al. (2009).



(a) General Sum (b) Fully Cooperative (c) Zero-Sum Policy Error (d) Zero-Sum Value Error

Figure 12. Mean policy and value error of stochastic fictitious play using different learning rate schedules in 10^5 randomly generated NFGs with different game theoretic properties. For the value error, ground truth values are computed using MSA with 10^7 iterations. Temperatures of the Logit equilibria are randomly sampled in the interval $[0, 10]$.

In Figure 12, we test the different learning rate schedules in randomly generated NFGs. For the random generation, we uniformly sample utilities of a 2-player NFG with 6 actions per agent. To test different game theoretic properties, we additionally perform experiments with normalized utilities according to the fully cooperative or zero-sum property. Then, we approximate the Logit equilibrium using a different budget of solver iterations. The policy error is calculated as the absolute difference in policy between two steps of SFP, i.e $|SBR(\pi_{-i}^t, \tau) - \pi_i^t|$. Additionally, we compute the value error in the zero-sum games, as they always have a unique Logit equilibrium (Hofbauer & Sandholm, 2002). In all experiments, the learning rate schedule of Nagurney and Zhang (1996) performed best.



(a) $\tau = 0$ (b) $\tau = 1$ (c) $\tau = 10$

Figure 13. Examples of the smooth best response dynamics at different temperatures for the matching pennies game. Both player have two actions and get a reward of +1 if they choose the same action and -1 otherwise. The Logit equilibria are marked in red.

In Figure 13, an example of the smooth best response dynamics is visualized at different temperatures for the matching pennies game. In the matching pennies game, both player have two actions and get a reward of +1 if they choose the same action and -1 otherwise. The Logit equilibrium is always unique for sufficiently small temperatures, but does not have to be for larger temperatures if multiple Nash equilibria exist (McKelvey & Palfrey, 1995). In this example, at $\tau = 0$ and $\tau = 1$, the LE is unique, but at $\tau = 10$ multiple LE exist, which approximate the three Nash equilibria of the game. Note that depending on the initialization, SFP will converge to different LE.

B.3. Quantal Stackelberg equilibrium

The computation requires finding a global optimum. We utilize a Dinkelbach-Type algorithm[2] (Cerny et al., 2020), which relies on fractional programming (Dinkelbach, 1967). The definition of a QSE can be reformulated as:

$$\pi_i = \underset{\tilde{\pi}_i \in \Delta_i}{\arg\max} \frac{\sum_{a \in A_{-i}} u_i(\tilde{\pi}_i, a) \exp(\tau\, u_i(\tilde{\pi}_i, a))}{\sum_{a \in A_{-i}} \exp(\tau\, u_i(\tilde{\pi}_i, a))}.$$

The primary notion of fractional programming is the transformation of the problem $\max_x f(x)/g(x)$ into a different problem $F(p) = \max_x f(x) - pg(x)$, maximizing the original problem at the root $F(p^*) = 0$. Since p is a scalar and F convex, one can find the global optimum using simple binary search. In each iteration, the binary search solves the Dinkelbach subproblem

$$\max_{\tilde{\pi}_i \in \Delta_i} \sum_{a \in A_{-i}} \big[u_i(\tilde{\pi}_i, a) - p\big] \exp(\tau\, u_i(\tilde{\pi}_i, a)).$$

Solving the subproblem requires finding a global optimum of a simpler problem than the original formulation, but the solution is still difficult to compute. In our games with small action spaces, it was sufficient to approximate the global optimum using grid search. Notably, there exist other methods using piece-wise linear approximation or gradient descent (Cerny et al., 2020).

C. AlphaZero in Competitive Simultaneous Games

The only component requiring change when adapting AlphaZero to simultaneous games is the tree search algorithm. We evaluate three different tree search variants, namely Monte-Carlo Tree Search (MCTS) (Metropolis & Ulam, 1949), Counterfactual Regret Minimization (CFR) and fixed depth search. MCTS is intended for sequential games, but there also exists an adaptation for simultaneous games, namely Simultaneous-Move Monte-Carlo Tree Search (SM-MCTS) (Bošanský et al., 2016). We refer to it simply as MCTS since only considering simultaneous games renders the distinction redundant. For MCTS, we evaluate three different selection functions.

The *Decoupled Upper Confidence bound for Trees (DUCT)* uses the standard Upper Confidence bound for trees like AlphaZero (Silver et al., 2018), but every player keeps an independent statistic of action-values and action-visits. Specifically, each agent selects a move according to

$$a^* = \underset{a_i \in A_i}{\arg\max} \frac{w_{a_i}}{n_{a_i}} + c\, \pi_\theta(o_i, a_i) \frac{\sqrt{N}}{n_{a_i}},$$

where w_a is the sum of values propagated through the current node in the backup phase and n_{a_i} the number of times agent i selected action a_i. N denotes the total number of visits in the current node and c is a parameter balancing exploration and exploitation. The policy $\pi_\theta(o_i, a_i)$ is used to guide the exploration of the tree search. This adaptation is simple to implement and has been shown to work well in a variety of games (Lanctot et al., 2013; Tak et al., 2014). However, it has also been shown that DUCT does not always converge to a Nash equilibrium (Shafiei et al., 2009). This is, because the action selection is deterministic and independently of the other agents. As a result, it is possible that all agents select the same actions over and over, leading to a cycle around the true Nash equilibrium. This problem can be alleviated by using a random tie break between actions with the same upper confidence bound. However, even with this extension, convergence cannot be guaranteed.

In contrast to DUCT, the *Exponential Weight Algorithm for Exploration and Exploitation (EXP3)* (Auer et al., 2002) is a stochastic selection algorithm. Again, all players select their action independently from each other, but they sample their action from the following distribution

$$\begin{aligned}\sigma(a_i) &= \frac{(1-\gamma)\exp(\eta w_{a_i})}{\sum_{\tilde{a_i} \in A_i} \exp(\eta w_{\tilde{a_i}})} + \frac{\gamma}{|A_i|} \\ &= \frac{(1-\gamma)}{\sum_{\tilde{a_i} \in A_i} \exp(\eta(w_{a_i} - w_{\tilde{a_i}}))} + \frac{\gamma}{|A_i|},\end{aligned}$$

[2]There are two errors in the original paper. In Algorithm 1, line 4 the subtraction needs to be an addition, and in line 7, the if and else cases need to be swapped.

where $\eta = \gamma/|A|$ and γ is an exploration parameter. The second formula is numerically more stable, because it avoids the computation of large exponential terms. In addition to the selection function, EXP3 also slightly alters the computation of the backup function:

$$w_{a_i} \leftarrow w_{a_i} + \frac{\tilde{w}}{\sigma(a_i)}.$$

The outcome of the current evaluation is scaled by the probability that an action is taken to account for very unlikely events. The final resulting policy of EXP3 is the average of the sample probabilities over all iterations. In contrast to DUCT, EXP3 has been proven to converge to a Nash equilibrium in Normal-form two-player zero-sum games (Auer et al., 2002).

In contrast to DUCT and EXP3, using *Regret Matching* an action is not selected according to the expected outcome of that action, but proportional to the expected regret of not choosing that action. Similar to EXP3, Regret Matching is a stochastic algorithm which samples an action from the distribution

$$\sigma(a) = \begin{cases} r_a^+/R^+ & R^+ > 0 \\ 1/|A| & \text{else} \end{cases}, \qquad R^+ = \sum_{\tilde{a} \in A} r_{\tilde{a}}^+,$$

where $(\cdot)^+ = \max(0, \cdot)$. The regret values r_a are computed during the backward pass for all actions that are not selected. To compute the regret, the algorithm needs to keep track of the average outcome of all joint actions a, not just the individual actions a_i like DUCT and EXP3. Like Exp3, Regret Matching always converges to a Nash equilibrium in Normal-form two-player zero-sum games (Lisy et al., 2013).

Counterfactual Regret Minimization (CFR) (Brown et al., 2019) is a tree search for games with imperfect-information. Since games with imperfect information are a superset of simultaneous perfect-information games, CFR can be applied. However, CFR is unnecessarily complicated and inefficient, because the only imperfect information arises from the simultaneous move selection. Specifically, a simultaneous perfect-information game can also be modeled as a sequential imperfect-information game, where agents do not know the selected move of the other agents. This leads to the property, that the value of a node in the search tree only depends on the subtree below, but not on the previous game dynamics. An algorithm exploiting this property is *Simultaneous Move Online Outcome Sampling (SM-OOS)* (Bošanský et al., 2016). Because CFR relies on Regret Matching, SM-OOS is very similar to MCTS with Regret Matching as a selection function. However, there are a few important differences. In SM-OOS, only one player updates their regret values in each iteration, while in MCTS with Regret Matching all players update their regrets. The updating agent explores the state space by choosing a random action with probability ϵ or playing on policy with probability $1 - \epsilon$. The non-updating agents play on policy to ensure that the regret calculation of the updating player is correct. Additionally, the updating agent keeps track of its tail and sampling probability. The tail probability is the product of the policy action probabilities of all nodes from the current node to the leaf node on the path taken during action selection. The sampling probability is very similar, but uses the action probabilities including exploration instead of the plain policy. By weighting the regret updates with the ratio of tail and sampling probability, one can ensure that the computed regret accurately reflects the regret that would occur when the agent plays on policy. Lastly, only the non-updating player adds their current policy estimates to the cumulative policy sum at each node to prevent a mixture with the exploration probability. Even though the calculation with a single updating agent is more accurate than the computation in MCTS, it is also less efficient as less updates happen in the same computation time.

Fixed depth search, also called Backward induction (Rust, 2018), is an algorithm originally intended for solving a complete game tree, but it is also possible to use the algorithm on a truncated game tree with a heuristic evaluation in the leaf nodes (Bošanský et al., 2016). Firstly, the game tree is built up to a specific depth. Then, all leaf nodes are evaluated. Lastly, the values of the leaf nodes are propagated upward the game tree to the root node using a backup function. We test two backup functions, which are based on the idea of solving for an equilibrium. Using the game theoretic algorithms presented in Appendix B, we test a backup function based on the Nash equilibrium as a Logit equilibrium with fixed temperature ($\tau = 10$).

C.1. Baseline Agent

In Battlesnake, area control is a standard heuristic for evaluating a game state (Schier & Wüstenbecker, 2019). In the stochastic game modes, area control implicitly incentivizes a snake to eat more food than the enemy, because it can reach more squares if it is able to win a head-to-head collision. We compute area control using a flood-fill algorithm (Pavlidis, 1982), which fills the board starting from the heads of all living snakes. If two snakes are able to reach a grid square at the

same time, we use the length as a tie break according to the head-to-head collision rule. In the stochastic game modes, our variant of flood fill also dynamically deletes the current tail of all snakes in every iteration. This simulates the game dynamics as every snake would move forward and leave the square occupied by its tail. The only exception is a situation, where a snake has just eaten a food in the last turn. Then, the tail of this snake stays on its square in the first iteration of flood fill and is only deleted in the second iteration.

As an additional improvement, we combine the area control of each snake with their relative health score to prevent them from starving. Let b^2 be the total number of grid squares, $\tilde{N}$ the current agents alive, h_{max} the maximum health, h_i the health of agent i and α_i their area control computed as described above. Then, we evaluate a board position for player i as

$$\tilde{w}_i = \frac{1}{2}(\tilde{\alpha}_i + \tilde{h}_i - \frac{1}{|\tilde{N}|}\sum_{j \in \tilde{N}} \tilde{h}_j),$$

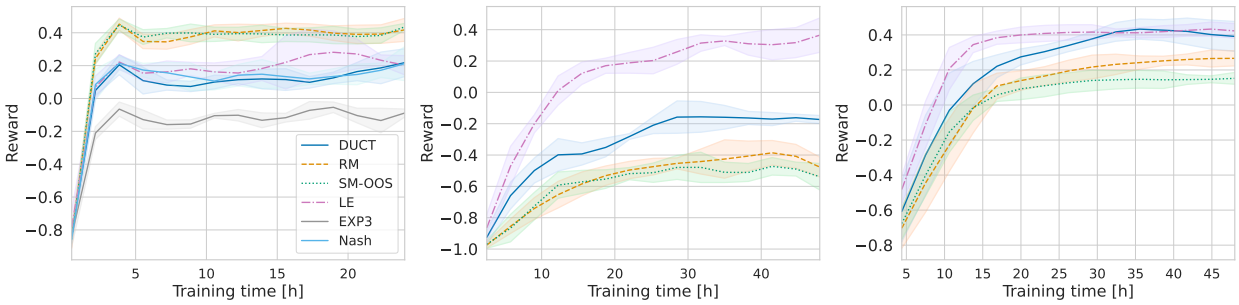
where $\tilde{\alpha}_i$ is the area control advantage of agent i relative to the board size and $\tilde{h}_i$ their relative health. Specifically, these values are computed as

$$\tilde{\alpha}_i = \frac{1}{b^2}(\alpha_i - \frac{1}{|\tilde{N}|}\sum_{j \in \tilde{N}} \alpha_j), \qquad \tilde{h}_i = \frac{h_i}{h_{max}}.$$

For Tron, we omit the terms using a health score and simply evaluate a game state using $\tilde{\alpha}_i$. To compute a policy from the heuristic value function, we utilize MCTS with DUCT as a selection function. For DUCT, we use the standard exploration bonus of $c = \sqrt{2}$. Since the baseline agent does not have a trained policy model to guide the search, we omit the policy guidance term in DUCT.

C.2. Evaluation

We evaluate the different tree search variants in the three game modes of Battlesnake against the baseline agent using $2e3$ search iterations. For all game modes, we train AlphaZero with the adapted tree search on five seeds. The game of Tron has short episodes and a smaller state space, such that we limit training time to a single day. In contrast, for the stochastic game modes, we train AlphaZero for two days.



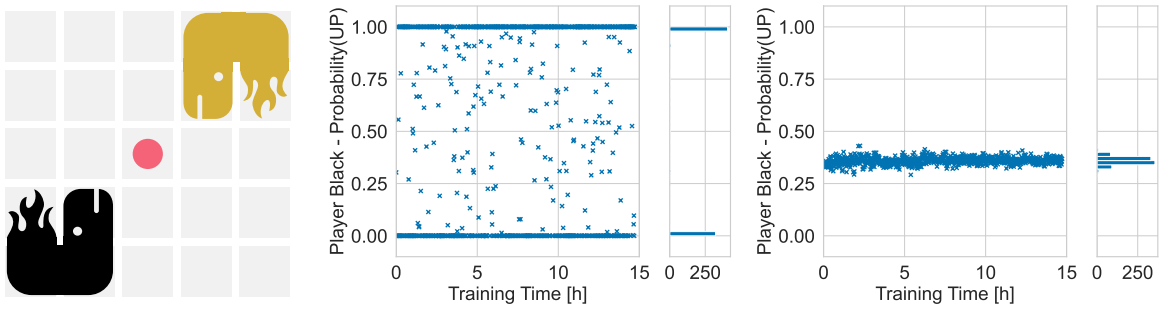
(a) Tron (b) Stochastic 2 Player (c) Stochastic 4 Player

Figure 14. Evaluation of AlphaZero with different Search variants in sim. games. Mean and standard deviation are computed over five seeds.

In Figure 14, the evaluation results are displayed. Note that we excluded the worst performing variants, namely MCTS with Exp3 selection function and fixed depth search with Nash equilibrium backup, from the experiments of the stochastic modes to save resources. In Tron, RM and SM-OOS perform best. However, in the stochastic mode with two players, LE outperforms all other methods by a large margin. In the mode with four players, LE achieves the best results as well, but closely followed by DUCT. Overall, we conclude that fixed depth search with Logit equilibrium backup is the best adaptation of AlphaZero to simultaneous perfect-information games.

D. Stability of a Unique Learning Target

Even though the Logit equilibrium inherently includes an error probability, it may be preferable to a Nash equilibrium as it is more stable. As an example, consider a situation where an agent has two possible actions. If both actions have the same expected utility, selecting either action or any probability distribution over both actions is a Nash equilibrium. However, if one of those actions has a marginally higher expected utility, this action is assigned a probability of one. As a result, a Nash equilibrium may be sensitive to small approximation errors in the utility function. Those errors, however, are always present when using neural networks as value function approximations.



(a) Game Situation (b) Nash Equilibrium (c) Logit Equilibrium

Figure 15. Comparison of equilibria in a single game situation using multiple training checkpoints. The policy of the Nash- and Logit equilibrium for an NFG generated by the value function of a neural network at different training time is compared.

We empirically investigate this property in the non-deterministic mode of Battlesnake with a board of size 5×5. Using AlphaZero with DUCT, we train a neural network for 15 hours on a single RTX3090 GPU and 15 Intel Xeon Gold 6258R CPU cores. During training, we save checkpoints of the model weights in regular time intervals. Afterwards, we compute the Nash equilibrium and Logit equilibrium for a single fixed board position, which is displayed in Figure 15. This board position has the property of inhibiting multiple actions with equal expected utility, i.e. the actions up and right for the black agent. For the Logit equilibrium, we use a temperature of $\tau = 5$. We plot the probability of the black agent playing action up in the Nash- or Logit equilibrium. Additionally, we compute the histogram of probabilities for this action over the training time. Evidently, the value function of the neural network changes during training, such that the probability oscillates between zero and one for the Nash equilibrium. However, the action probabilities of a Logit equilibrium for the same neural network are much more stable. This is due to the entropy based smoothing, which prevents a strong oscillation. We believe this to be an indication Logit equilibria are a better target for updating a neural network than Nash equilibria.

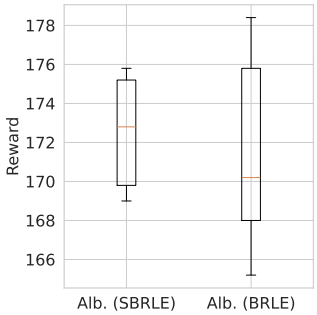


Figure 16. Comparison of Albatross trained with SBRLE and BRLE with the behavior cloning agent in the Cramped Room layout. Starting from the same proxy model, we trained response models using BRLE and SBRLE on five different seeds.

This stability of Logit equilibria also translates to a greater stability of SBRLE than BRLE. As an ablation study, we train

the response model of Albatross using BRLE in the Cramped Room layout of Overcooked. Event though there is only a small difference, Albatross with a SBRLE performs better than with BRLE and has less variation in the results.

E. Maximization of Transformed Utilities with Shannon Entropy

We present a mathematical proof that the softmax function maximizes smooth best responses with entropy regularization. This is a well known fact in literature and easy to verify, but to the best of our knowledge the proof has never been published.

Theorem E.1. *The transformed utilities $\tilde{u}_i(\pi)$ defined as $\tilde{u}_i(\pi) = u_i(\pi) + \frac{1}{\tau}\psi(\pi_i)$ using Shannon entropy $\psi(\pi_i) = \sum_{a_i \in A_i} \pi_i(a_i) \log(\pi_i(a_i))$ as a smoothing function are maximized by the softmax function $\pi_i = SBR(\pi_{-i}, \tau) \propto \exp(\tau\, u_i(\,\cdot\,, \pi_{-i}))$.*

Proof. We define the theorem as a maximization problem and solve it with the method of Lagrange multipliers. The objective function f is

$$\begin{aligned} f(\pi_i) &= u_i(\pi_i, \pi_{-i}) + \frac{1}{\tau}\psi(\pi_i) \\ &= \left[\sum_{a_i \in A_i} \pi_i(a_i)\, u_i(a_i, \pi_{-i})\right] - \frac{1}{\tau} \sum_{a_i \in A_i} \pi_i(a_i) \log(\pi_i(a_i)), \end{aligned}$$

where the policy π_{-i} of the other agents is fixed. We use $u_i(a_i, \pi_{-i})$ as a shorthand for $u_i(\hat{\pi}_{a_i}, \pi_{-i})$ with $\hat{\pi}_{a_i}$ being the policy that assigns probability one to action a_i and zero otherwise. The policy π_i has to be a valid probability distribution, which yields the constraint

$$g(\pi_i) = -1 + \sum_{a_i \in A_i} \pi_i(a_i) = 0.$$

The resulting Lagrangian function is defined as $L(\pi_i, \lambda) = f(\pi_i) + \lambda g(\pi_i)$. Setting the derivative of the Lagrangian function L with regards to the policy of an action $\pi_i(a_i)$ to zero results in

$$\begin{aligned} 0 &= \frac{\partial L}{\partial\, \pi_i(a_i)} \\ &= \frac{\partial f}{\partial\, \pi_i(a_i)} + \lambda \frac{\partial g}{\partial\, \pi_i(a_i)} \\ &= u_i(a_i, \pi_{-i}) - \frac{1}{\tau}\Big(1 + \log\big(\pi_i(a_i)\big)\Big) - \lambda. \end{aligned}$$

We can directly solve this equation for the policy $\pi_i(a_i)$, which yields

$$\begin{aligned} \pi_i(a_i) &= \exp\big(\tau\, u_i(a_i) - \tau\lambda - 1\big) \\ &= C\, \exp\big(\tau\, u_i(a_i, \pi_{-i})\big). \end{aligned}$$

Therefore, it holds that $\pi_i(a_i) \propto \exp\big(\tau\, u_i(a_i, \pi_{-i})\big)$, since $C = \exp(-\tau\lambda - 1)$ is a constant. □

F. Concavity of Likelihood Function

We prove the concavity of the likelihood function by showing that the second derivative is always less or equal to zero. For an alternative proof, we refer to the work of McFadden (1973).

Theorem F.1. *Given K observations of actions $(a_i^1, \ldots, a_i^K)$ and corresponding optimal policies $(\pi_{-j}^1, \ldots, \pi_{-j}^K)$ of the other agents, the log-likelihood function l for the temperature τ of agent i is concave.*

Proof. The second derivative of the likelihood function is:

$$\frac{\partial^2 l}{\partial^2 \tau} = -\sum_{k=1}^{K} \Bigg[\frac{\Big(\sum_{a_i \in A_i} \big(u_i^k(a_i, \pi_{-i}^k)\big)^2 \exp\big(\tau\, u_i^k(a_i, \pi_{-i}^k)\big)\Big)\Big(\sum_{a_i \in A_i} \exp\big(\tau\, u_i^k(a_i, \pi_{-i}^k)\big)\Big)}{\Big(\sum_{a_i \in A_i} \exp\big(\tau\, u_i^k(a_i, \pi_{-i}^k)\big)\Big)^2} - \frac{\Big(\sum_{a_i \in A_i} u_i^k(a_i, \pi_{-i}^k) \exp\big(\tau\, u_i^k(a_i, \pi_{-i}^k)\big)\Big)^2}{\Big(\sum_{a_i \in A_i} \exp\big(\tau\, u_i^k(a_i, \pi_{-i}^k)\big)\Big)^2} \Bigg].$$

We continue to show that the term in square brackets is greater equal zero for all $k \in \{1, \dots, K\}$. The denominator of the term is always positive and can therefore be ignored in our analysis.

$$\begin{aligned}
& 0 \leq \Big(\sum_{a_i \in A_i} \big(u_i^k(a_i, \pi_{-i}^k)\big)^2 \exp\big(\tau\, u_i^k(a_i, \pi_{-i}^k)\big)\Big)\Big(\sum_{a_i \in A_i} \exp\big(\tau\, u_i^k(a_i, \pi_{-i}^k)\big)\Big) \\
& \qquad - \Big(\sum_{a_i \in A_i} u_i^k(a_i, \pi_{-i}^k) \exp\big(\tau\, u_i^k(a_i, \pi_{-i}^k)\big)\Big)^2 \\
\Leftrightarrow \quad & 0 \leq \sum_{a_i \in A_i} \sum_{b \in A_i} \big(u_i^k(a_i, \pi_{-i}^k)\big)^2 \exp\big(\tau\, u_i^k(a_i, \pi_{-i}^k)\big) \exp\big(\tau\, u_i^k(b_i, \pi_{-i}^k)\big) \\
& \qquad - \sum_{a_i \in A_i} \sum_{b_i \in A_i} u_i^k(a_i, \pi_{-i}^k) u_i^k(b_i, \pi_{-i}^k) \exp\big(\tau\, u_i^k(a_i, \pi_{-i}^k)\big) \exp\big(\tau\, u_i^k(b_i, \pi_{-i}^k)\big) \\
\Leftrightarrow \quad & 0 \leq \sum_{a_i \in A_i} \sum_{b_i \in A_i} \big(u_i^k(a_i, \pi_{-i}^k)\big)^2 - u_i^k(a_i, \pi_{-i}^k)\, u_i^k(b_i, \pi_{-i}^k) \\
\Leftrightarrow \quad & 0 \leq 2 \sum_{a_i \in A_i} \sum_{b_i \in A_i} \big(u_i^k(a_i, \pi_{-i}^k)\big)^2 - u_i^k(a_i, \pi_{-i}^k)\, u_i^k(b_i, \pi_{-i}^k) \\
\Leftrightarrow \quad & 0 \leq \sum_{a_i \in A_i} \sum_{b_i \in A_i} \big(u_i^k(a_i, \pi_{-i}^k)\big)^2 + \big(u_i^k(b_i, \pi_{-i}^k)\big)^2 - 2\, u_i^k(a_i, \pi_{-i}^k)\, u_i^k(b_i, \pi_{-i}^k) \\
\Leftrightarrow \quad & 0 \leq \sum_{a_i \in A_i} \sum_{b_i \in A_i} \big(u_i^k(a_i, \pi_{-i}^k) - u_i^k(b_i, \pi_{-i}^k)\big)^2.
\end{aligned}$$

As all terms of the sum are greater equal zero, the sum itself is greater equal zero. Therefore, the second derivative of the log-likelihood function is negative. Consequently, the log-likelihood function is concave. □

The line search algorithm for computing the maximum likelihood for the temperature of weak agent j is displayed in Algorithm 4.

Algorithm 4 Maximum Likelihood Estimation of Temperature τ_j

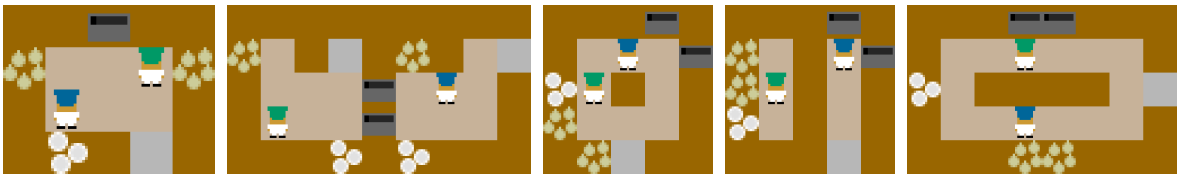
Input: Observations $(a_j^1, \dots, a_j^K)$, ground truth policies $(\pi_{-j}^1, \dots, \pi_{-j}^K)$, valid temperature interval $[\tau_{max}, \tau_{min}]$, iteration number m
for $i = 0$ **to** $m - 1$ **do**
  $\tau \leftarrow \frac{1}{2}(\tau_{max} + \tau_{min})$
  if $\frac{\partial l}{\partial \tau} > 0$ **then**
    $\tau_{min} \leftarrow \tau$
  else
    $\tau_{max} \leftarrow \tau$
  end if
end for
Output: $\tau_j = \frac{1}{2}(\tau_{max} + \tau_{min})$

The algorithm iteratively reduces the interval of possible temperature estimations by computing the sign of the gradient at the midpoint of the interval. If the gradient is positive, the maximum likelihood estimate has to be in the upper half of the interval, otherwise the lower half.

For Algorithm 4, we assume that optimal ground truth policies $(\pi^1_{-j}, \ldots, \pi^K_{-j})$ for agents $-j$ are given. If there exists a unique Nash equilibrium, then the optimal policies are well defined as the equilibrium policies. However, computing a Nash equilibrium in games with many agents is difficult and most games do not have a unique Nash equilibrium. As a solution, one can substitute the ground truth policies with the policy of the Logit equilibrium of temperature τ_{max} learned by the proxy model, i.e $\pi_{\theta_P}(o_i, \tau_{max})$. Consequently, the bounded rationality of the weak agents is no longer modeled according to globally optimal play, but rather relative to the rationality acquired during training. Additionally, this implicitly solves the equilibrium selection process, because rationality is modeled relative to the learned equilibrium during training.

G. The Game of Overcooked

Cooperation performance in the game of Overcooked is evaluated in five different layouts, introduced by Carrol et al. (2019). The layouts are displayed in Figure 17. In the cramped room layout, agents have little space to move around and need to focus on good movement coordination. In the asymmetric advantage layout, one agent has a much shorter path between onion dispenser and cooking pot, while the other agent has a shorter path between serving location and cooking pot. Therefore, agents should distribute tasks in a way that takes into account these advantages. The distribution of tasks can be seen as high-level coordination, while the coordination of movement in the cramped room layout is low-level coordination. In the coordination ring layout, agents have a single circle of free squares available. They have to coordinate the direction of movement, such that they do no block each others way. In the forced coordination layout, only one agent has access to onion and dish dispenser, while the other agent has access to cooking pots and serving locations. Consequently, the first agent has to pass onions and dishes over a counter to the second agent. This makes training much more difficult, because the first agent only receives a reward signal if the second agent is sufficiently trained to use the tools or ingredients received. But, to train the second agent, the first agent needs to pass them ingredients and tools, which rarely happens through random play. In the counter circuit layout, the agents have a single circuit available for movement, similar to the coordination ring layout. However, the circle is longer, which forces the agents to pass onions over the middle counter to achieve a perfect score. All layouts have a fixed starting position, which are displayed in Figure 17. For evaluation, we swap starting positions every other episode.



(a) Cramped Room (b) Asymmetric Advantage (c) Coord. Ring (d) Forced Coord. (e) Counter Circuit

Figure 17. Different kitchen layouts in the cooperative game of Overcooked as introduced by Carrol et al. (2019). The Cramped Room and Coordination Ring layout test low-level coordination since the agents should not block each other. In the Asymmetric Advantage and Counter Circuit layout high-level coordination, i.e. distribution of tasks, is tested. In the Forced Coordination layout, cooperation performance is limited by the weakest of the two agents since both need to perform distinct tasks.

In Overcooked, multiple implementations or versions with vastly different levels of difficulty exists. For example, as of January 2024, the newest version of the implementation of Carrol et al. (2019) allows agents to start cooking a soup even if not all ingredients are present in the soup. This produces a soup, which does not yield any rewards when serving. However, previous versions did not allow this behavior, which made solving the environment vastly easier. Additionally, very early versions of Overcooked had an option to use a dense shaped reward based on the distance to pots or dispensers. Newer versions still allow to specify these rewards, but this specification is internally ignored. Since these versions developed over time, it is difficult to determine which version was used in the experiments of different researchers. For our experiments, we opted to use the game dynamics used by Lou et al. (2023) as their training scheme PECAN was previously state of the art in Overcooked. That is, we disallowed the premature cooking of a soup with missing ingredients and did not use any distance-based rewards. To increase the training speed, we reimplemented the game of Overcooked in C++, which also provides an environment with fixed game dynamics for reproducibility in future work.

Another challenge when comparing results is the usage of a behavior cloning agent as a proxy for human behavior. These agents are trained on a small dataset of human play. However, we hypothesize that during recording the human actions, the time between environment steps was too short for the human contestants to react. That is, because most actions (58%) in the human dataset are the "do nothing" action. Consequently, the trained behavior cloning agents exhibit a behavior of staying at a single point most of the time. In literature, different workarounds for this issue were used. Carrol et al. (2019) used the agents as is, but implemented a dynamic into the game of Overcooked which forces agents to take a random action if they have been stuck at one position for more than three time steps. This dynamic intends to free agents if they are stuck at a position and block each other. In addition to this dynamic, Lou et al. (2023) filtered the probability distribution predicted by the trained agent, completely removing the stay action. In our implementation, we again used the same setup as Lou et al. (2023). To facilitate reproduction of our results, we publish our trained human behavior cloning agents.

H. Hyperparameters

To support reproducibility of our results, we report all hyperparameters used in our experiments. In this section, we list common hyperparameters used across all experiments. Hyperparameters, which differ between experiments are listed in Table 2. For gradient updates, we use the AdamW optimizer (Loshchilov & Hutter, 2017) with a weight decay factor of $1e{-}5$. We anneal the learning rate using a cosine decay from $1e{-}3$ to $1e{-}6$. The loss function consists of a value and policy term, which are summed. The value term is the mean squared error between prediction and target, while the policy term is the cross entropy between predicted and target policy. For the neural network, we use the MobileNetV3 architecture (Howard et al., 2019). During backup of tree search, a Logit equilibrium is solved at every non-leaf node. To solve the Logit equilibrium, we perform 150 iterations of Stochastic Fictitious Play (Hofbauer & Sandholm, 2002) with the step size annealing of Nagurney and Zhang (1996), as discussed in Appendix B.2. During training, we balance exploration and exploitation using Boltzmann-Exploration (Cesa-Bianchi et al., 2017) with a probability of 0.5 or sample from the LE-policy at the root node with equal probability of 0.5. In all experiments, we trained our models on five seeds. Unless specified otherwise, we used Nvidia RTX3090 GPU and 14 Intel Xeon Gold 6258R CPU for each GPU. Those numbers were chosen to optimally saturate the compute cluster used.

Table 2. Hyperparameters of Albatross in Overcooked and the different game modes of Battlesnake. Only hyperparameters that differ between modes are listed here.

Hyperparameter	Overcooked	Tron	stoch. 2p. BS	stoch. 4p. BS	4p. Coop. Tron
Buffer Size	5e5	1e5	2e6	2e6	2e6
Number GPU (RTX 3090)	2	3	3	2	1
Search Depth	1	3	3	1	1
Total Training Time	48h	24h	96h	48h	24h
Discount Factor	0.9	0.99	0.99	0.99	0.97
Batch Size	15000	2000	2000	2000	12000

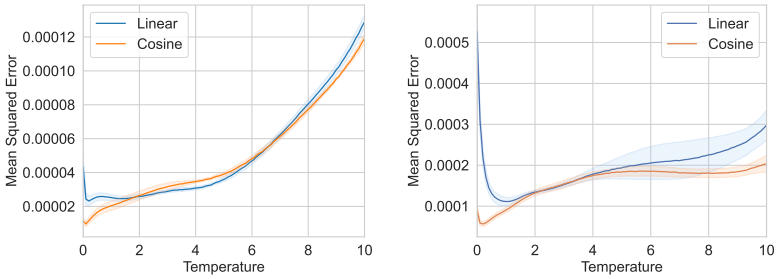
I. Additional Analysis of Albatross

We perform additional experiments to analyze the behavior and hyperparameters of Albatross. Firstly, we test the effect of different temperature distributions during training. Then, we perform additional analysis on the effect of the temperature when cooperating with different opponents.

I.1. Temperature Training Distribution

During training, we randomly sample a temperature from a fixed interval $[\tau_{min}, \tau_{max}]$. This sampling has the effect that the entire range of temperatures are learned, as well as all combinations of different temperatures. However, when using a neural network for function approximation, its predictions may degrade at the boundary of the input space, i.e. the interval of temperatures. Therefore, we bias the random sampling toward the upper and lower boundary of the interval of valid temperatures. Specifically, we use a cosine function to sample more often near the boundaries.

We compare the cosine based sampling against the uniform (linear) sampling in the deterministic zero-sum Battlesnake game mode with a board of size 5×5. Using this smaller board size, it is possible to compute the ground truth Logit equilibrium. This ground truth Logit equilibrium is unique for all temperatures, since the game is a two-player zero-sum



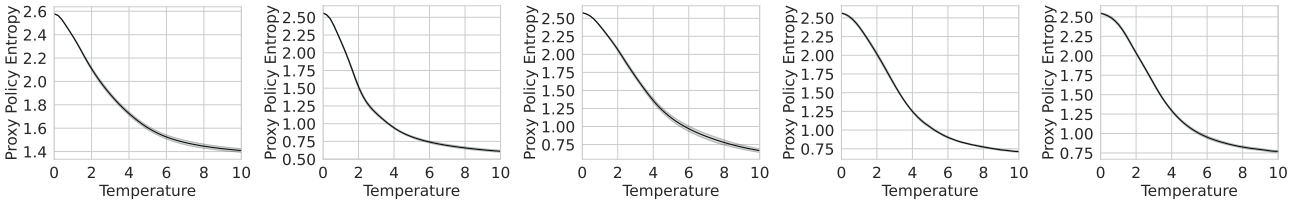
(a) Policy Function (b) Value Function

Figure 18. Mean squared error of the policy and value function of the proxy model in deterministic 2-player Battlesnake with a board size of 5×5. Since the game is zero-sum, it has a unique Logit equilibrium for all temperatures, which can be computed exactly due to the small state space.

game. In Figure 18, the mean squared error of the policy and value function is displayed. For most parts of the temperature input space, the cosine sampling produces a lower error than the linear sampling. Especially for small temperatures near zero, the linear sampling produces larger error spikes than cosine sampling. Consequently, we use the cosine function in all of our experiments.

I.2. Temperature Estimation

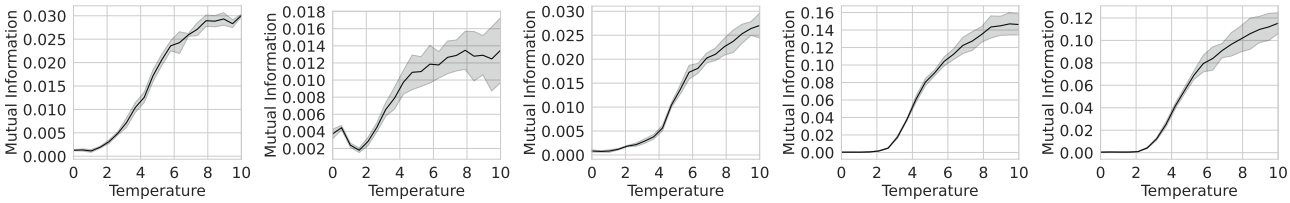
Following the experiments presented in Section 5, we present the full experimental data on all layouts of Overcooked and in all game modes of Battlesnake.



(a) Cramped Room (b) Asym. Advantage (c) Coord. Ring (d) Forced Coord. (e) Counter Circuit

Figure 19. Entropy of the proxy model policy at different temperatures for all five layouts in the game of Overcooked.

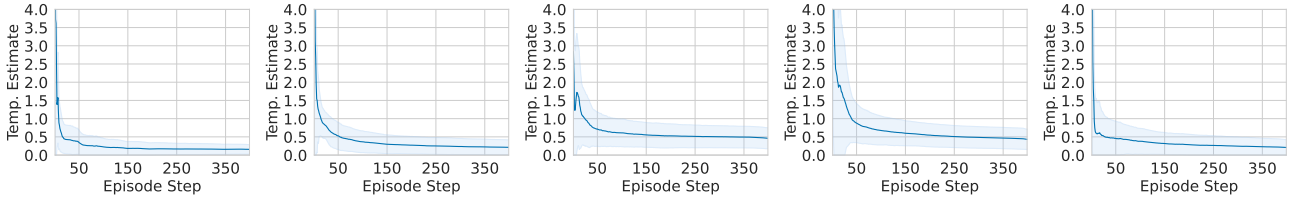
In Figure 19, the entropy of the proxy policy at different temperatures in displayed for the different layouts in Overcooked. In all layouts, for temperature $\tau = 0$, the entropy starts at about $\log_2(6) = 2.58$, which is the entropy of a uniformly random policy with six discrete actions. The entropy decreases at higher temperatures and converges to the entropy of the learned Logit equilibrium at highest temperature $\tau = 10$, which may be different for every layout.



(a) Cramped Room (b) Asym. Advantage (c) Coord. Ring (d) Forced Coord. (e) Counter Circuit

Figure 20. Mutual information between Albatross response and proxy model at different temperatures for all five layouts in the game of Overcooked.

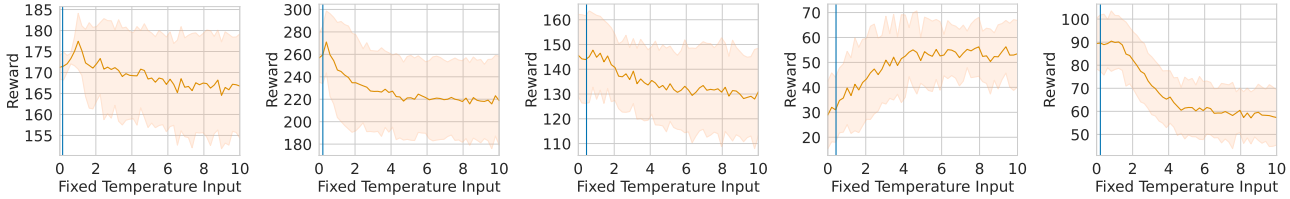
In Figure 20, the mutual information between proxy and response model at different temperatures are displayed. For a temperature of $\tau = 0$, the mutual information in all game modes is close to zero, indicating that the agents do not cooperate. This is expected, since the proxy model at this temperature only plays a uniformly random policy. At high temperatures, the mutual information between the agents rises as they start to cooperate. The mutual information can also be seen as a measure of trust from the response model, that the proxy model fulfills their expected task in cooperation.



(a) Cramped Room (b) Asym. Advantage (c) Coord. Ring (d) Forced Coord. (e) Counter Circuit

Figure 21. Temperature estimation of Albatross for the human behavior cloning agent per time step in the different layouts of Overcooked.

In Figure 21, we display the result of the Maximum Likelihood estimation of Albatross when playing the human behavior cloning agent. For details regarding the MLE, we refer to Algorithm 4. In all layouts, after only a few episode steps, the temperature estimation converges toward the true temperature. This indicates, that it is possible to estimate the rationality of an agent within a single episode of Overcooked. However, as discussed in Section 6, we see that the MLE needs between 10 and 30 episode steps to reach a stable estimation, which may not be feasible in very short games.



(a) Cramped Room (b) Asym. Advantage (c) Coord. Ring (d) Forced Coord. (e) Counter Circuit

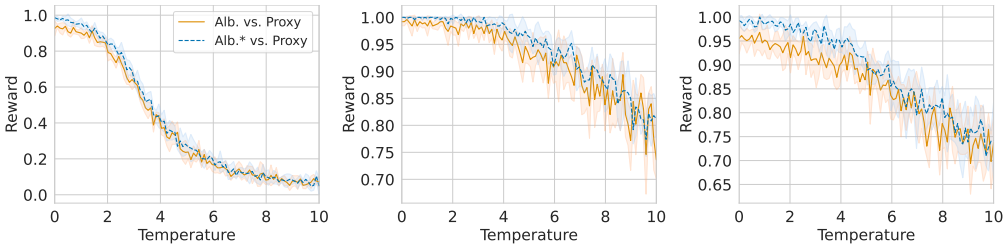
Figure 22. Analysis of the robustness of Albatross to a wrong temperature estimation in all layouts of Overcooked.

In Figure 22, we test the robustness of Albatross to wrong temperature estimations in the different layouts of Overcooked. To this end, we evaluate Albatross with a fixed temperature input against the human behavior cloning agent. For all layouts, except Forced Coordination, the highest reward is achieved when using a temperature close to the true temperature. In those layouts, the achieved reward drops when using temperature estimations far from the ground truth temperature. In the Forced Coordination layout, the training process at low temperatures is much more difficult, because only sparse rewards resulting from the nearly random proxy policy are observed. Therefore, Albatross performs better at higher temperature estimations. Note that the reported reward does not necessarily align perfectly with the reward achieved by MLE, because in some situations the adaptive estimation of MLE may be advantageous. For example, the human behavior cloning agent performs better in some parts of the state space and as a consequence the best temperature estimation differs between episodes.

In Figure 23, we test the performance of Albatross against the proxy model in the game of Battlesnake at different temperatures, akin to Figure 7. At low temperatures, the proxy model plays nearly random and consequently the response model wins nearly all games. In contrast, at high temperatures the proxy model plays rationally and the reward achieved by the response model drops. We denote Albatross with fixed ground truth temperature input as Albatross*, and compare its performance to standard Albatross with MLE. In the game of Battlesnake, the reward difference between MLE and given ground truth is small, indicating that MLE is able to accurately estimate rationality.

I.3. Behavior of Albatross in Repeated Matrix Games

The behavior of Albatross and (S)BRLE as well as the temperature estimation can be explained using simple repeated matrix games. For simplicity we use the BRLE, because SBRLE would require the computation of a softmax that only obfuscates



(a) Tron (b) Stochastic 2 Player (c) Stochastic 4 Player

Figure 23. Evaluation of Albatross versus the proxy model at different temperatures.

the explanation. Depending on the response temperature used, the results may vary, but the main points of this explanations are also applicable to the SBRLE. Let's suppose we have the matrix of a Normal form game, which is given in Table 3.

Table 3. Example Normal form game.

	P2, A1	P2, A2
P1, A1	4, 4	0, 0
P1, A2	1, 1	2, 2

Pareto optimal play would be the Nash equilibrium at (A1, A1) with utility of 4 for both players. Suppose P1 plays according to a BRLE and estimates the temperature of P2, who plays according to some specific pattern. For example, if P2 always plays A1, then their rationality is estimated as positive infinity (which we clip to some maximum temperature), since they always play the best action. Hence, the best response to a Logit equilibrium of maximum temperature would be A1 for P1. This dynamic is illustrated in the left part of Table 4.

Table 4. Behavior of Albatross with Pareto optimal player (left) and Tit-for-Tat agent (right)

Time Step	1	2	3	...	1	2	3	4	5	...	k
P1, Action	1	1	1	...	1	2	2	2	2	...	2
P2, Action	1	1	1	...	2	1	2	2	2	...	2
Temp. Estimate	-	Max	Max	...	-	Min	0	0.17	0.27	...	Max
Utility	4	4	4	...	0	1	4	4	4	...	4

In contrast, if P2 always plays A2, then the temperature estimate is negative infinite (clipped to some minimum temperature). Therefore, both players would continue to play A2, which is the other pure Nash equilibrium (A2, A2) of the game.

Now suppose P2 plays an adaptive "Tit for Tat" strategy, which always copies the last action of P1. Depending on the initial actions chosen by both players, different behaviors emerge. We use an optimistic initialization for BRLE, choosing A1 (corresponding to high rationality estimation of P2) at first. If P2 also plays A1 at first, then both players will continue to play A1 forever (see first table). In contrast, if P2 starts with A2, then the initial rationality estimate would be the minimum clipped temperature. Then, P2 copies A1 from the first turn. Having played A1 and A2 both once, the rationality estimate is zero as it corresponds to uniform random play. Then, both agents continue to play A2 and the temperature estimate rises to the maximum clipped temperature. This behavior can be seen in the right part of Table 4. We should note, that Albatross was not designed to play adaptive agents. Instead, we make the standard assumption of opponent modelling that the other agents play a static strategy only depending on the environment state.