# THE GOOD SHEPHERD: AN ORACLE AGENT FOR MECHANISM DESIGN

**Jan Balaguer, Raphael Köster, Christopher Summerfield & Andrea Tacchetti**
DeepMind, UK
{jua, rkoster, csummerfield, atacchet}@deepmind.com

## ABSTRACT

From social networks to traffic routing, artificial learning agents are playing a central role in modern institutions. We must therefore understand how to leverage these systems to foster outcomes and behaviors that align with our own values and aspirations. While multiagent learning has received considerable attention in recent years, artificial agents have been primarily evaluated when interacting with fixed, non-learning co-players. While this evaluation scheme has merit, it fails to capture the dynamics faced by institutions that must deal with adaptive and continually learning constituents. Here we address this limitation, and construct agents ("mechanisms") that perform well when evaluated over the learning trajectory of their adaptive co-players ("participants"). The algorithm we propose consists of two nested learning loops: an inner loop where participants learn to best respond to fixed mechanisms; and an outer loop where the mechanism agent updates its policy based on experience. We report the performance of our mechanism agents when paired with both artificial learning agents and humans as co-players. Our results show that our mechanisms are able to shepherd the participants strategies towards favorable outcomes, indicating a path for modern institutions to effectively and automatically influence the strategies and behaviors of their constituents.

## 1 INTRODUCTION

Modern institutions often serve two distinct and equally important roles in our society. First they mediate and foster economic or social interactions among citizens (e.g. taxation policies ensure governments receive enough funds to build roads and schools). Second, they foster behaviors that bring us closer to our aspirations as a society (e.g. charitable donations are tax-free). As artificial learning agents mediate more and more interactions among humans, firms, and organization, it is paramount that we study how to construct adaptive systems that can fulfil both roles.

However, while multiagent learning has received considerable attention in recent years, the standard evaluation scheme pairs our artificial agents with other fixed, and potentially adversarial co-players (e.g. exploitability) (Vinyals et al., 2019; Muller et al., 2019; Goodfellow et al., 2014). While this evaluation scheme has merit, it fails to capture the dynamics faced by modern institutions that are often paired with learning constituents, and where agents must take into account not only what other agents will do next, but also, in the long run, how they will adapt to the current strategies present in the system.

Here we address this shortcoming and construct low-exploitability agents that do well when paired with learning co-players, in the general-sum setting. We construct players that, through their behavior, are able to influence what others will learn to do, and explicitly leverage the link between one agent's actions and another agent learning trajectory. In other words, we construct agents ("mechanisms") that learn to act so as to shepherd participants' strategies both at equilibrium, and during learning.

Our proposed method takes the form of an inner-outer loop learning process. In the inner loop, participant agents respond to a fixed mechanism strategy, while in the outer-loop our mechanism agent adapts its policy based on experience. Unlike previous work, our mechanisms make very few assumptions on the preferred strategies, outcomes, or learning capabilities of the participants, and only have access to the consequences of their own behavior on the learning of others.

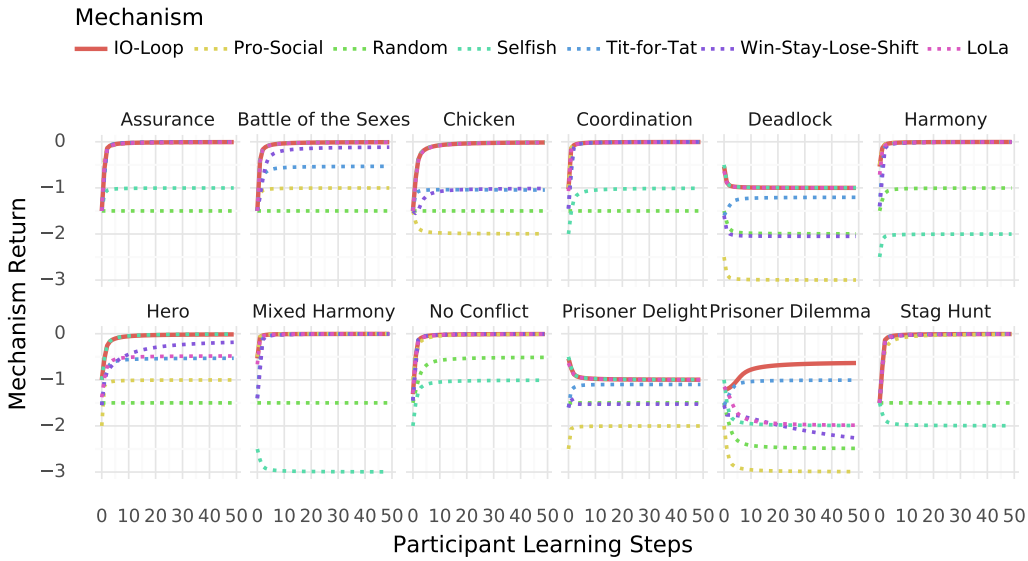


Figure 1: Performance of fixed mechanisms paired with adaptive participants in the 12 matrix games we consider. The horizontal axis shows the learning steps of the participant agent, while the vertical axis shows the return collected by the fixed mechanism agent. Our IO-loop agent, here trained with Diff-MD, tracks the best alternative strategy in all games but Prisoner Dilemma where it significantly outperforms it. Plots are produced averaging mechanism returns over 5 random seeds.

We investigate the performance of our mechanism agents with both artificial and human co-players in simple 2-player 2-strategy repeated games, and in a stylized resource allocation problem. We study how our method can be adapted when mechanisms are granted access to the inner workings of participants, and when that is not the case.

Our results show that our mechanism agents successfully shepherd the learning of others towards desirable outcomes, and that the direction presented here is promising for agent-agent interactions, and withstand a transfer to the agent-human interaction setting.

In the broader context of AI in modern day institutions, our methods and ideas show that adaptive agents can successfully shepherd the learning of their co-players towards desirable outcomes and behavior, opening the door to learning-based institutions that fine-tune the incentives faced by their constituents in pursuit of group level goals.

## 2 Related work

The inner-outer loop method that we propose here provides insights into two challenges for multiagent reinforcement learning.

The first challenge is the non-stationarity of the environment. When training multiple policies simultaneously, the environment is non-stationary from the point of view of any agent due to the change in the other players' policies. Common approaches to mitigate this non-stationarity involve building populations of agents (Brown, 1951; Muller et al., 2019; Vinyals et al., 2019) or exploiting knowledge of the learning dynamics of others (Balduzzi et al., 2018; Hemmat et al., 2020). Both these approaches have often focused on competitive zero-sum game. Here we focus on setting where the environment is always stationary from the point of view of all agents since we don't update policies concurrently in the same training loop.

Second is the challenge of *equilibrium selection*. Generally, and particularly in non-zero-sum games, multiple Nash Equilibria may exist. This leads to the problem of both finding and selecting among possibly unequal equilibria, with the goal of a) biasing learning towards outcomes preferred by one agent (shaping), or b) generalizing with unseen co-players. Progress towards this has been

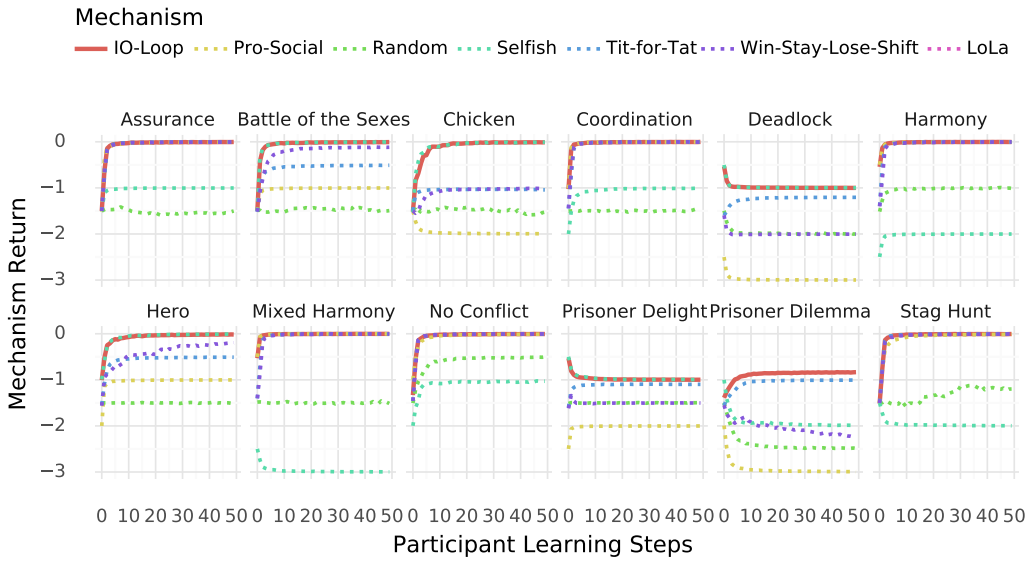


Figure 2: Performance of fixed mechanisms paired with adaptive participants in the 12 matrix games we consider. The horizontal axis shows the learning steps of the participant agent, while the vertical axis shows the return collected by the fixed mechanism agent. Our IO-loop agent, here trained with ES-MD, tracks the best alternative strategy in all games but Prisoner Dilemma where it significantly outperforms it. Plots are produced averaging mechanism returns over 5 random seeds.

made in recent years through *centralized learning with decentralized execution*, a framework for multiagent RL where agents can exploit privileged knowledge about other agents' during training, but not at time of deployment. Centralized learning can be useful for equilibrium selection: for example, access to a centralized value function provides a recipe to construct agents that are able to coordinate at execution in cooperative settings (Sunehag et al., 2017); coupled training of multiple agents can improve learning of communication protocols (Foerster et al., 2016; 2019); learning from interactions with agents at different stages of training can improve generalization at evaluation with human participants (Strouse et al., 2022); and exploiting information about how other agents update their behaviour can be used to shape them, both within an episode (Lerer & Peysakhovich, 2018; Peysakhovich & Lerer, 2018) and across training (Foerster et al., 2017; Yang et al., 2020). In our method, we do not separate training from execution. We make no assumptions about the learning rule of the co-players, or how they will adapt to the strategies currently present in the system. Instead, we infer the relationship between the mechanism's actions and the participant's learning directly from the observed interactions.

## 3 METHODS

We consider the problem of constructing agents ("mechanisms") that shepherd the learning of others ("participants"). We use repeated symmetric two-player, two-strategies games as our initial test-bed as they are easy to analyze and train on. We then move on to a simple resource allocation game. We start by assuming that the mechanism agent has access to the inner workings of participant agents in Differentiable mechanism design, and then extend our methods to remove this assumption using Evolutionary Strategies (Salimans et al., 2017).

### 3.1 ENVIRONMENTS

We tackle iterated 2-player, 2-strategies symmetric matrix games, and a simple resource allocation game with one mechanism agent and four participant agents.

### 3.1.1 ITERATED MATRIX GAMES

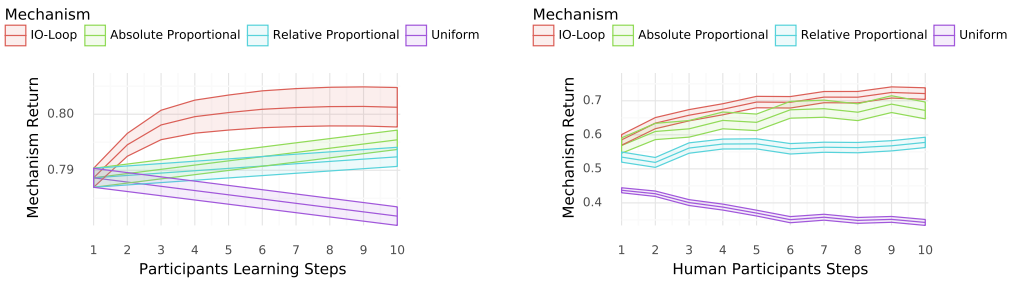


Figure 3: Performance of fixed mechanisms paired with artificial adaptive participants (left) or human participants (right) in the resource allocation game we consider. The horizontal axis shows the learning steps of the participant agents (independent learning), while the vertical axis shows the return collected by the fixed mechanism agent. Our IO-loop agent, here trained with Diff-MD, outperforms alternative mechanisms proposed in the economics literature for this game both with artificial learning and human co-players. The left panel was produced averaging mechanism returns over 50 random seeds, while the right panel shows the average return over each experiment; shaded areas indicate standard error.

We consider the 12 symmetric 2-player, 2-strategies matrix games identified in (Wikipedia, 2022; Robinson & Goforth, 2005), with payouts scaled down by 4 (our payouts are between -3 and 0), and consider iterated interactions between a mechanism (row) player and a participant (column) player with a single memory step. Our naming convention is most easily followed when focusing on the Prisoner Dilemma game.

For each game we define a Markov Decision Process with statespace $\mathcal{S} = (s_0, CC, CD, DC, DD)$, with $s_0$ being the initial state, and the rest being the state after the joint actions in the previous state (e.g. cooperate, cooperate; cooperate, defect and so on). Our agents' one-memory policies can be represented by a 5-tuple $\theta$ corresponding to the probability of cooperating on each state. In these simple repeated games, the transition kernel $\mathcal{T}$ takes the form of a matrix whose entries describe the probability of the next state as a function of the previous state, and can be derived analytically given the one-memory policy parameters from both players. The reward functions are specified on arrival at each state as $r_m = (0, r_R, r_S, r_T, r_P)$ and $r_p = (0, r_R, r_T, r_S, r_P)$, where $r_P$, $r_R$, $r_S$, $r_T$ correspond to the punishment, reward, sucker and temptation payoffs respectively. The returns $R_m$ and $R_p$ that both the mechanism and participant aim to maximize corresponds to the state value for the initial state $s_0$.

### 3.1.2 RESOURCE ALLOCATION GAME

We further consider a modification of the classic Public Goods Game (as described in (Koster et al., 2022)). The game consists of a single interaction between four participants $i = 1, 2, 3, 4$ and a single mechanism agent. Each participant receives an endowment $e_i$ and allocates a fraction $\rho_i$ of it to a common investment pool, which is then grown by a fixed constant factor $(1.6)$ and redistributed to participants in full. The specific amount received by each participant $i$ is denoted as $p_i$ and is determined by the mechanism. Participants seek to maximize their individual welfare (i.e. $R_{p_i} = p_i + (1 - \rho_i)e_i$), while the mechanism seeks to maximize total participants' welfare (i.e. $R_m = \frac{1}{N}\sum_i^N R_{p_i}^t$, with $N$ the number of participants). When interacting with naive or poorly designed redistribution mechanisms, the incentive to free-ride may tempt each player away from the contributing to the common pool, which in turn decreases total welfare.

## 3.2 THE INNER OUTER LOOP ALGORITHM

Our learning process takes the form of an inner-outer loop that exposes the mechanism to the consequences of its actions on the learning of others. In the inner loop, participant agents repeatedly interact with a fixed mechanism, and use independent gradient ascent to improve their own policies. In the outer loop, mechanism agents update their strategies based on the experience they acquired in the inner loop.

**Algorithm 1** Inner outer loop algorithm. Given an underlying Game function ($\mathcal{MDP}$) that takes policy parameters $\theta_m$ and $\theta_p$ for the mechanism and participants respectively computes their returns $R_m$ and $R_p$, a random participant parameter generator $\mathfrak{R}$, and initial mechanism parameters $\theta_m^0$ this algorithm produces a policy $\theta_m^{T_m}$ for the mechanism player.

**Require:** $\mathcal{MDP}, T_m, T_p, \mathfrak{R}, \theta_m^0, \gamma_m, \gamma_p$
  **for** $t_m$ in $0 :: T_m$ **do**
    $\theta_p^0 \sim \mathfrak{R}$;  $\bar{R}_m \leftarrow 0$
    **for** $t_p$ in $0 :: T_p$ **do**
      $(R_m, R_p) \leftarrow \mathcal{MDP}(\theta_m^{t_m}, \theta_p^{t_p})$
      $\bar{R}_m \leftarrow \bar{R}_m + R_m$
      $\theta_p^{t_p+1} \leftarrow \theta_p^{t_p} + \gamma_p \nabla_{\theta_p^{t_p}} R_p$
    **end for**
    $\theta_m^{t_m+1} \leftarrow \theta_m^{t_m} + \gamma_m \nabla_{\theta_m^{t_m}} \bar{R}_m$
  **end for**

#### 3.2.1 Differentiable mechanism design

We first consider the case where the mechanism agents can directly compute the gradients of its return $\bar{R}_m$ with respect to its policy parameters $\theta_m$. Inspecting Algorithm 1 we note that, at any given $\theta_m^{t_m}$ update, the mechanism return $R_m$ depends on the mechanism parameters $\theta_m^{t_m}$, as well as on the entire trajectory of participants parameters over the inner loop $\theta_{p_0}, \ldots, \theta_p^{T_p}$. In Differentiable Mechanism design (Diff-MD), we let the mechanism update have gradient access to the entire trajectory, as well as the environment transition kernel $\mathcal{T}$. In practice, this can easily be implemented using a tensor library with auto-differentiation (such as JAX (Bradbury et al., 2018)).

#### 3.2.2 Evolutionary strategies for non-differentiable mechanism design

When the mechanism agent cannot take derivatives through the environment, we used evolutionary strategies (ES). In this case, the inner loop is repeated $N_p$ times so as to form an experience "batch" with mechanism parameters slightly perturbed at the beginning of each inner loop ($t_p = 0$) as $\theta_m^p = \theta_m^p + \epsilon_p$, where $\epsilon_p \sim \mathcal{N}(0, \sigma_m^2)$ with $\sigma_m$ a hyper-parameter (1 in our experiments). Given an experience batch the mechanism policy gradient, estimated as $\nabla_{\theta_m} \approx \sum_{p=1}^{N_p} \frac{\epsilon_p \bar{R}_m}{N_p \sigma_m}$, moves the mechanism parameters in the direction of those used in episodes that led to positive outcomes (see (Salimans et al., 2017) for details). We refer to this method as ES-MD.

### 3.3 Learning with Opponent-Learning Awareness (LOLA)

We implemented LOLA (Foerster et al., 2017) as a baseline in our experiments. In LOLA, the mechanism agent projects the learning of the participants forward in time. In contrast to the original paper, in which both agents are assumed to be using LOLA, here only let the mechanism agent be learning-aware.

## 4 Results

Here we show how a trained mechanism performs when paired with learning participants. We report the return collected by a (fixed) mechanism in each episode over the learning trajectory of its co-players. Figures 1 and 3 show how mechanisms trained with Diff-MD perform in the 12 matrix games and resource allocation game respectively, while Figure 2 shows the performance of a mechanism trained with ES-MD in the 12 matrix games we consider.

### 4.1 Matrix games

In the matrix games we compare our mechanism (labelled as IO-Loop in the legends) to well known one memory strategies (e.g. Tit-for-Tat) and pure strategies (e.g. Selfish). Additionally, in Diff-MD we also compare a mechanism trained with LOLA. In all games, and for both Diff-MD and

ES-MD, our IO-loop mechanism achieves the same performance as the best alternative available strategy, with the exception of Prisoner Dilemma where it significantly outperforms it. We used the following hyper-parameters for both experiments: $T_m = 10000, T_p = 50, \gamma_m = 0.1, \gamma_p = 10, \theta_m^0 = [0.5, 0.5, 0.5, 0.5, 0.5]$, in the ES-MD experiment we further set $N_p = 256$ and $\sigma_m = 1$.

### 4.2 Resource allocation game

In the resource allocation game we report mechanism performance when paired both with artificial learning agents and human co-players. In particular, in Fig. 3, we consider unequal endowments with $e_i \in [0.2, 1.0]$, and compare our mechanism with four alternative redistribution strategies: Absolute Proportional and Relative Proportional redistribute funds proportionally to the absolute contribution $\rho_i e_i$ or to the fraction of endowment $\rho_i$ contributed by each participants, while the Uniform and Random mechanisms redistributed the funds equally and randomly respectively. Figure 3 shows that Diff-MD finds a mechanism policy that shepherds the participants towards higher welfare outcomes. In this experiment we represented the participants policy as their propensity to contribute to the public fund: $\theta_{p_i} = \rho_i$, and the mechanism policy as a MLP with a single 32-units hidden layer. We further set $T_m = 5000, T_p = 10, \gamma_m = 0.01, \gamma_p = 0.1$ and $\theta_m^0$ the default MLP initialization.

### 4.3 Evaluation with human co-players

Figure 3 (right) shows the performance of our mechanism, and alternative baselines, when paired with human co-players in the stylized resource allocation game outlined above (endowment condition for the 4 players: $[1.0, 0.5, 0.4, 0.3]$). We used crowd-sourcing platforms to collect data, and all participants gave informed consent to participate in the experiment. Participants were organized in groups of 4, and after a tutorial phase, they played the resource allocation game outlined above completing the 10 steps constituting our "inner loop". The tutorial round explained the mechanics of the game, instructed participants on how to use the web interface, and outlined how participants would be rewarded in real money: participants received a base compensation for completing the experiment, and a bonus proportional to their aggregate return over the course of the experiment. After one game with the Uniform mechanism, each group of participants interacted with two mechanisms, either resulting from our training, or with one of the baseline mechanisms outlined above in counterbalanced order. We collected data from 236 non overlapping groups. If a participant dropped out during the experiment, their actions were replaced with random actions, which were subsequently removed in the analysis ($39\%$ of responses were removed this way). The results presented in the right hand panel in Figure 3 show that our mechanism could withstand a basic transfer to interacting with human co-players, and that its performance remained consistent with what we observed in simulation.

## 5 Conclusion

We have shown here that our inner-outer loop algorithm can provide an oracle-style benchmark to test agents' ability to shepherd the behavior of learning co-players to desired outcomes, and that in simple environments, our agents transfer to human co-players.

As more and more of the systems we use and deploy become adaptive, it becomes increasingly important to 1) construct agents that can plan and act taking into account the fact that others are learning around them, and 2) construct agents that can shape the incentives faced by co-players in pursuit of group-wide objectives.

The ideas and results presented here show that exposing agents to the consequences of their actions on the learning of others is a sensible first step toward these goals. Moreover, the transfer to human co-players we were able to showcase suggests that our method contains the basic elements required to design adaptive institutions can fulfill their basic "mechanical" mediation function in society, as well as shepherd their constituents towards more desirable strategies and behaviors.

## REFERENCES

David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. In *International Conference on Machine Learning*, pp. 354–363. PMLR, 2018.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

Jakob Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 1942–1951. PMLR, 2019.

Jakob N Foerster, Yannis M Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*, 2016.

Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*, 2017.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

Reyhane Askari Hemmat, Amartya Mitra, Guillaume Lajoie, and Ioannis Mitliagkas. Lead: Least-action dynamics for min-max optimization. *arXiv preprint arXiv:2010.13846*, 2020.

Raphael Koster, Jan Balaguer, Andrea Tacchetti, Ari Weinstein, Tina Zhu, Oliver Hauser, Duncan Williams, Lucy Campbell-Gillingham, Phoebe Thacker, Matthew Botvinick, and Christopher Summerfield. Human-centered mechanism design with democratic ai, 2022.

Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning, 2018.

Paul Muller, Shayegan Omidshafiei, Mark Rowland, Karl Tuyls, Julien Perolat, Siqi Liu, Daniel Hennes, Luke Marris, Marc Lanctot, Edward Hughes, et al. A generalized training approach for multiagent learning. *arXiv preprint arXiv:1909.12823*, 2019.

Alexander Peysakhovich and Adam Lerer. Consequentialist conditional cooperation in social dilemmas with imperfect information, 2018.

D. Robinson and D. Goforth. *The Topology of the 2x2 Games: A New Periodic Table*. Routledge advances in game theory. Routledge, 2005. ISBN 9780415336093. URL `https://books.google.co.uk/books?id=HEacmydRoNcC`.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017.

DJ Strouse, Kevin R. McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data, 2022.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Wikipedia. Normal-form game — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Normal-form%20game&oldid=1045662802`, 2022. [Online; accessed 01-February-2022].

Jiachen Yang, Ang Li, Mehrdad Farajtabar, Peter Sunehag, Edward Hughes, and Hongyuan Zha. Learning to incentivize other learning agents. *arXiv preprint arXiv:2006.06051*, 2020.