Ranking Joint Policies in Dynamic Games using Evolutionary Dynamics

Natalia Koliou
University of Piraeus
Piraeus, Greece
nataliakoliou@iit.demokritos.gr

George Vouros
University of Piraeus
Piraeus, Greece
georgev@unipi.gr

ABSTRACT

Game-theoretic solution concepts, such as the *Nash equilibrium*, have been key to finding stable joint actions in multi-player games. However, it has been shown that the dynamics of agents' interactions, even in simple two-player games with few strategies, are incapable of reaching *Nash equilibria*, exhibiting complex and unpredictable behavior. Instead, evolutionary approaches can describe the long-term persistence of strategies and filter out transient ones, accounting for the long-term dynamics of agents' interactions. Our goal is to identify agents' joint strategies that result in stable behavior, being resistant to changes, while also accounting for agents' payoffs, in dynamic games. Towards this goal, and building on previous results, this paper proposes transforming dynamic games into their empirical forms by considering agents' strategies instead of agents' actions, and applying the evolutionary methodology *α-Rank* to evaluate and rank strategy profiles according to their long-term dynamics. This methodology not only allows us to identify joint strategies that are strong through agents' long-term interactions, but also provides a descriptive, transparent framework regarding the high ranking of these strategies. Experiments report on agents that aim to collaboratively solve a stochastic version of the graph coloring problem. We consider different styles of play as strategies to define the empirical game, and train policies realizing these strategies, using the DQN algorithm. Then we run simulations to generate the payoff matrix required by *α-Rank* to rank joint strategies.

KEYWORDS

Evolutionary Dynamics, Empirical Games, Stochastic Games, Deep Reinforcement Learning, Ranking Strategy Profiles

ACM Reference Format:
Natalia Koliou and George Vouros. 2025. Ranking Joint Policies in Dynamic Games using Evolutionary Dynamics. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Game theory studies agents' strategies not only in terms of optimality of performance but also with regard to stability of agents' behavior. Game-theoretic solution concepts, particularly the *Nash equilibrium*, have played an important role in this research. However, solution concepts do not account for the long-term dynamics of agents' interactions, which are important in dynamic settings. In static games, where payoff matrices are known, studying solution concepts is relatively straightforward. For example, the mixed strategy *Nash equilibrium* for the Rock-Paper-Scissors game shown in Table 1 occurs when both players randomize their choices uniformly across Rock, Paper, and Scissors. Accounting for the dynamics of agents' interactions over time in dynamic settings, we need to analyze agents' behavior in terms of their payoffs, identifying joint strategies that result into agents' stable behaviors. Evolutionary approaches have shown great potential towards this aim.

	Rock	Paper	Scissors
Rock	0,0	-1,1	1,-1
Paper	1,-1	0,0	-1,1
Scissors	-1,1	1,-1	0,0

Table 1: Payoff matrix for the Rock-Paper-Scissors game.

To study agents' behavior in multi-agent dynamic settings, researchers often train deep learning models to learn joint policies. These models, either in collaborative or competitive settings, are usually trained with the ultimate objective to result into Nash equilibria, aiming to agents' stability of behavior, where no agent has an incentive to deviate from their joint policy. In complex dynamic settings with long-term dynamics of agents' interactions, there is no guarantee of reaching that objective and there is no way to reveal the reasoning behind the agents' choice of a policy instead of another. Although proposals towards explainability and interpretability of models are important, these aim to provide either explanations for the policy as a whole (i.e. agent's style of play) or about individual decisions. In our case, we need a descriptive framework to account for transparency regarding the strength of agents' policies, accounting for long-term dynamics.

Our goal is to identify agents' individual policies that result in stable (i.e., resistant to changes) behavior while playing with others, accounting for long-term agents' interactions and agents' payoffs, in dynamic games. This is motivated by the need to identify strategies of human or software agents that need to act as co-players in a common setting, considering that these agents have policies that have been formed independently: This is in contrast to assuming that agents have been trained to learn a joint policy. We conjecture that using a descriptive evolutionary framework approach helps agents select strong (i.e. non-transient) policies when playing with or against other agents in dynamic settings. Identifying these policies and understanding their "superiority" is important, particularly in collaborative scenarios where agents must choose

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

their own strategies while interacting with others that use specific styles of problem solving.

Towards this goal, we propose exploiting multiple policy models, each realizing a distinct style of play (*strategy*), and then defining an empirical game for evaluating agents' joint performance when they play jointly using various strategy profiles. This empirical game is exploited by the *α-Rank* evolutionary framework [13] to evaluate the evolutionary dynamics of agents' strategies over time, ultimately identifying which ones prevail in the long run.

Although this work builds on the *α-Rank* framework, it contributes a perspective for evaluating individual agents' strategies in stochastic, sequential decision making settings, when they act with other agents following specific styles of play.[1] In so doing, we:

- Describe a concise methodology for evaluating and ranking agents' policies, accounting for their long-term interactions in dynamic settings, using the *α-Rank* evolutionary framework.
- Demonstrate this methodology in multi-agent graph coloring dynamic games, defining multiple styles of play (i.e. strategies) per agent.
- Show how agents' choices of strategies—and the policies realizing them—can be transparently justified by means of a descriptive framework.

2 BACKGROUND

In this section, we outline the key concepts necessary to follow the proposed approach.

2.1 Dynamic Games

Dynamic games describe agent interactions along the time dimension. Unlike static games, where players execute single one-shot actions, dynamic games involve a series of decisions made by each of the players at subsequent points in time. A key property of dynamic games is that the actions taken at any given moment influence the future states of the system and future decisions made. These temporal dependencies require players to consider the long-term consequences of their actions. A dynamic game can be represented as a tuple $G = (S, K, A, T, P)$, where S represents a finite set of states, K is the set of players, and $A = (A^k \times A^{-k})$ is the set of joint actions, where A^k corresponds to the set of actions available to the player k. A^{-k} denotes the set of actions available to players other than k. The transition function T describes the dynamics of the setting, determining the next state of the system based on the current state and the actions chosen by the players. Finally, $P^k : S \times (A^k \times A^{-k}) \times S \to \mathbb{R}$ is the payoff function for player k, given the current joint state, the action chosen by player k and the actions of the other agents, and the resulting state.

In this work we focus on stochastic dynamic games, as introduced by L.S. Shapley in 1953 [16]. In stochastic games, the outcome of players' actions is influenced by probabilistic events, making future states of the game uncertain. These games are often referred to as Markov games [17]. Therefore, in stochastic games, the transition function T is defined as a probability distribution over next states. Specifically, $T : S \times A \to \Delta(S)$, where $\Delta(S)$ is a probability distribution over the states, given a state and joint action. For example, in poker, while players' actions do influence the outcome, the next state of the game also depends on luck, such as drawing a strong hand like a flush or a weak hand like a pair of twos. In such games, players, when planning their actions, must account for both the actions of their opponents and the dynamics of the environment.

In dynamic games, players aim to decide on the course of their joint actions through time (joint policy) to maximize their accumulated rewards over time:

$$\sum_t T(s_t, (a_t^k, a_t^{-k}), s_{t+1}) \cdot P^k(s_t, (a_t^k, a_t^{-k}), s_{t+1})$$

Here, T specifies the transition probability from state s_t to the state s_{t+1} given (a_t^k, a_t^{-k}), and $P^k(s_t, (a^k, a^{-k}), s_{t+1})$ is the reward the player receives for choosing action a^k, given the actions a^{-k} of the other players, at state s_t, and resulting into state s_{t+1}.

2.2 Empirical Analysis and Empirical Games

Empirical Game Theory Analysis (EGTA) provides a framework that uses empirical methods to analyze player interactions within complex game environments [10]. These methods are used to define game components, such as payoff matrices, based on observed interactions, rather than relying on predefined rules. Simulation is one such method, where agents repeatedly play a game, and payoffs are collected based on the outcomes of these interactions. Other techniques include sampling, where a subset of the action space is explored to approximate the payoffs for a wider set of actions, and machine learning methods to identify players' behavior and estimate outcomes based on historical data [27]. Empirical techniques are applied in cases where the action space is too large and complex to define manually, making payoff matrices impossible to generate from simple rules and assumptions.

An empirical game, also referred as a meta-game, is a *Normal Form Game* of the form $G = (K, \mathcal{S}tr, P)$, where K, $\mathcal{S}tr$ and P specify players, players' strategies, and payoffs, correspondingly. We define empirical games by abstracting the actions and defining the payoffs of players in an underlying dynamic game. The underlying game represents the actual setting where players interact. In the empirical game representation, K is the same as in the underlying game, i.e. the set of players engaged in strategic interactions. Strategies (i.e. styles of playing the game) in empirical games offer an action abstraction and can be derived by identifying distinct behaviors during game-play. The strategy space $\mathcal{S}tr$ consists of distinct agents' styles of play. $\mathcal{S}tr^k$ denotes the strategies of agent k and $\mathcal{S}tr^{-k}$ the set of strategies of agents other than k. The set of strategy profiles, i.e. agents' joint strategies, is defined to be $\mathcal{SP} = \{\mathcal{S}_i | \mathcal{S}_i = (str_i^1, str_i^2, \ldots, str_i^K)\}$, where $str_i^k \in \mathcal{S}tr^k$, and $i = 1, \ldots$ the profile index$\}$. The payoff matrix of an empirical game can be generated using empirical analysis techniques. Here, we focus on simulation, where agents engaged in the underlying game act according to policies adhering to specific strategies.

Subsequently, we use the terms *action* and *policy* when speaking about the underlying game, and the term *strategies* or *styles of play* when speaking about the empirical game.

[1]The source code is publicly available at: https://github.com/nataliakoliou/Collaborative-Graph-Coloring-Game.

The payoff function P of the empirical game is computed from simulations for each strategy profile as follows:

$$P^k(str^k, str^{-k}) = \frac{1}{N} \cdot \sum_{i=1}^{N} P_i^k(str^k, str^{-k})$$

where N is the number of simulation runs, str^k represents player k's strategy, str^{-k} denotes the strategies of the other players, and $P_i^k(str^k, str^{-k})$ (with an abuse of notation) represents the payoff player k receives in simulation run i when playing strategy str^k against the strategies of the other players. It must be noted that in contrast to dynamic games the payoff function does not take states as arguments, as the outcomes are determined by agents' joint strategies, i.e. $P^k : (\mathcal{S}tr^k \times \mathcal{S}tr^{-k}) \rightarrow \mathbb{R}$ [13]. If we aggregate these expected payoffs into a matrix, we get the empirical payoff matrix whose dimensionality is $\prod_{k=1}^{K} \mathcal{S}tr^k$. Each entry represents the expected payoff for strategy str^k against strategy str^{-k}.

2.3 The α-Rank Method

Evolutionary dynamics studies how agents' interactions in multi-agent settings evolve over time. While single-agent systems have acquired a strong foundation over the years [3], multi-agent systems are more challenging to analyze.

Current literature indicates a growing interest in studying the evolutionary dynamics of multi-agent systems [3] [6] [14]. In the context of games, evolutionary algorithms are widely used to explore game-theoretic concepts, resulting to the *Evolutionary Game Theory*. Building on work done in this area, *α-Rank* [13] introduces a novel game-theoretic approach to provide insights into the long-term dynamics of agents' interactions.

α-Rank is an evolutionary methodology designed to evaluate and rank agents' strategies in large-scale multi-agent interactions, introducing a new dynamic solution concept called *Markov-Conley chains* (MCCs). Given a K-player game, *α-Rank* considers the empirical game with K player slots, called *populations*, where individual agents correspond to strategies, i.e. to styles of playing the underlying game. Populations of agents interact with each other through an evolutionary process following the dynamics of games. The rewards received from these interactions determine how well each strategy performs and, in turn, how often it is adopted by individuals in the populations. Strategies that perform well have a higher probability of being adopted and carried over to the next generation, while those performing poorly are less likely to be adopted. This process leads to the evolution of populations.

To facilitate evolution, *α-Rank* uses the concept of mutation. Initially, populations are monomorphic, meaning all individuals within them choose the same strategy. During K-wise interactions, individuals have a small probability of mutating into different strategies or choosing to stick with their current one. The probability that the mutant will take over the population, defined to be the fixation probability function ρ, depends on the relative fitness of the mutant and the population being invaded. Fitness is a function that computes the expected reward an individual can receive when adopting a particular strategy, given the strategies of the other individuals. The stronger the fitness, the more likely it is for individuals to mutate, whereas the lower the fitness, the more likely it is for the mutant to go extinct. When the mutation rate is small, we can assume that the fitness for any agent k is $f^k(str^k, str^{-k}) = P^k(str^k, str^{-k})$, where P is the empirical game payoff.

Formally, the probability of a mutant strategy str' fixating in some population where individuals play strategy str is given by:

$$\rho_{str \rightarrow str'} = \frac{1 - e^{-\alpha \cdot \Delta f}}{1 - e^{-\alpha \cdot m \cdot \Delta f}} \tag{1}$$

assuming that Δf is non-zero. $\Delta f = f^k(str', str^{-k}) - f^k(str, str^{-k})$ represents the difference in fitness between the mutant strategy str' and the resident strategy str in the focal population k, while the remaining $K-1$ populations are fixed in their monomorphic strategies str^{-k}. Parameter m is the population size and α is the selection intensity. This adjusts the sensitivity of the system to fitness differences: with higher values of α, even small differences in fitness lead to larger changes in ρ. The nominator measures the potential of the mutant to "invade" the resident population solely based on its fitness advantage. Note that, for example, as Δf approaches zero, the probability of the mutant's success decreases. The denominator, on the other hand, normalizes the fixation probability using the population size m, making it more challenging for a mutant to dominate in larger populations. When Δf is zero, the fixation probability is equal to $1/m$ ([13], eq.13), indicating that the mutant strategy has the same probability of taking over as any other strategy in the population. We refer to this probability as the *neutral fixation probability*, denoted by ρ_m.

In the context of K-player games, *α-Rank* creates a Markov transition matrix over strategy profiles. This is an $|\mathcal{S}tr| \times |\mathcal{S}tr|$ matrix that defines the probability of moving from one strategy profile to another based on how likely each population is to change its strategy.

$$C_{str \rightarrow str'} = \begin{cases} \eta \cdot \rho_{str \rightarrow str'} & \text{if } str \neq str' \\ 1 - \sum_{str \neq str'} C_{str \rightarrow str'} & \text{otherwise} \end{cases} \tag{2}$$

Here, C is the strategy-transition matrix where each entry $C_{str \rightarrow str'}$ represents the probability of transitioning from strategy str to strategy str'. The first part of the formula calculates the probability of strategy transition, $\rho_{str \rightarrow str'}$ scaled by $\eta = \frac{1}{\sum_k (|Str^k|-1)}$, where k indexes populations. The second part of the formula computes the probability of staying with the same strategy str, excluding transitions to all other strategies.

This evolutionary process of competition and selection among players' strategies leads to a unique stationary probability distribution π of dimensionality $|\mathcal{SP}|$, where the mass assigned to a strategy profile indicates how likely it is to resist being "invaded" by other strategies as the dynamics evolve. To evaluate and rank strategy profiles —which is the ultimate goal— the method calculates π over the game's Markov chain, using the strategy-transition matrix C. This distribution indicates how often the system is likely to remain in each profile over time, allowing us to identify the most dominant strategies that are expected to prevail in the long run. Formally, π can be computed from the following equation:

$$\pi C = \pi \Rightarrow \pi(C - \mathbb{I}) = 0 \tag{3}$$

where $\mathbb{I}$ is the identity matrix. This means we are looking for a probability vector π such that when multiplied by the transition matrix C, it remains unchanged. To solve for π, the augmented matrix from $C - \mathbb{I}$ is constructed and a normalization condition to ensure that probabilities sum to 1 is imposed[2]. In this stationary distribution, $\pi = (\pi_1, \pi_2, ..., \pi_{|(SP)|})$, each π_i represents the average time the system spends in strategy profile i.

3 PROBLEM STATEMENT

As already stated, we aim at identifying (human and software) agents' strong joint strategies, in terms of stability and joint performance, to solve problems in dynamic settings, accounting for agents' long-term dynamics of interactions. Stability implies non-transient strong strategies, persisting in time, as they fit better to the objective of the agents given the structure of the game and payoffs received. However, in dynamic games, we need to define the payoff matrix and exploit this to determine strategies stability. Even if we manage to estimate payoffs, the computation of solution concepts like the *Nash equilibrium* imposes a high computational cost in these settings, does not guarantee convergence, and fails to scale to large games. Beyond identifying stable joint strategies, it is important to transparently justify/describe what makes one joint strategy better than another, providing evidence for the rankings.

We could, therefore, consider our problem as follows: Given a dynamic game G with K players, our goal is to identify styles of playing G, and thus, the set of strategy profiles $\mathcal{SP}$, and rank these profiles based on how stable they are over time, considering long-term agents' interactions towards achieving their objectives. Specifically, we aim to define a ranking function $\mathcal{R} : \mathcal{SP} \rightarrow \mathbb{R}$, where $\mathcal{R}(\mathcal{S}_i) > \mathcal{R}(\mathcal{S}_j)$ (resp. $\mathcal{R}(\mathcal{S}_i) \geq \mathcal{R}(\mathcal{S}_j)$) indicates that the strategy profile $\mathcal{S}_i$ is strictly (resp. weakly) preferred over $\mathcal{S}_j$, using a descriptive framework $\mathcal{D}$ defined over $\mathcal{SP}$, that provides transparency on how rankings are decided.

It must be noted that empirical game strategies are realized by agents' policies adhering to these strategies in the underlying game. Thus, identifying stable joint strategies in the empirical game translates to identifying stable joint policies adhering to these strategies in the underlying dynamic game.

4 PROPOSED METHOD

To address the challenge of identifying stable joint policies in dynamic games, we propose an approach that combines concepts from *Empirical Game Theory* and *Evolutionary Dynamics*, using *α-Rank*, providing transparency to rankings of agent's styles of play.

Given that the set of agents' policies in dynamic games can be infinitely large we focus on a subset of policies that adhere to concrete and well-defined styles of play. A way to identify styles of play is to observe how players behave in the underlying game or exploit demonstrations of game playing by means of style (mode) - preserving offline or inverse reinforcement learning methods. This may result into a mixture of policies (one per style of play) given that human experts performing a task usually follow a distinct set of specific styles based on well-established practices, preferences and experience. Having determined the game playing strategies, we can transform the dynamic game into its empirical form, defining the meta-game, as specified in Section 2.2: By (a) identifying empirical game strategies, and (b) training policies for agents to play the underlying game according to these strategies, (c) run policies to define the empirical game payoff matrix, through simulations.

Having defined the meta-game, we need to define the function $\mathcal{R}$, which ranks joint strategies based on agents' long-term dynamics and objectives. In our approach, we propose using the evolutionary *α-Rank* methodology to determine these rankings. The rankings are based on each strategy profile's evolutionary success, which is reflected in the probability of that profile being selected over time. This probability is captured by the stationary distribution π, which *α-Rank* computes in the limit of infinite ranking intensity α. As demonstrated by [13], a large α limit suffices. Therefore the long-term behavior is captured by the unique stationary distribution π under the large α limit. As it is proved in [13], the Markov chain associated with a generalized multi-population model, coincides with the MCC solution concept: MCCs can be identified effficiently in all games by the sink strongly connected components of a response graph, whose vertices correspond to pure strategies' profiles and directed edges from a strategy profile $\mathcal{S}_i$ to a strategy profile $\mathcal{S}_j$ specifies that $\mathcal{S}_j$ is weakly a better response than $\mathcal{S}_i$ for player k.

To compute π over strategy profiles, *α-Rank* requires the payoff matrix of the empirical game P. Along with the stationary distribution π, *α-Rank* outputs the fixation probability function $\rho_{\mathcal{S}_i \rightarrow \mathcal{S}_j}$, $\mathcal{S}_i, \mathcal{S}_j \in \mathcal{SP}$, which measures the likelihood of transitioning from one strategy profile $\mathcal{S}_i$ to another $\mathcal{S}_j$. Thus, *α-Rank* can be abstracted as a function:

$$\alpha\text{-Rank}(P) \rightarrow (\pi, \rho) \tag{4}$$

While the stationary distribution π provides valuable insight into the long-term behavior of strategies, it alone does not help us fully understand how strategies transition between one another. The fixation probability function ρ fills this gap. Based on this, the descriptive framework $\mathcal{D}$ can be adequately represented by π and ρ, which are constituents of the response graph that provides a complete view of the empirical game dynamics.

Overall, building on the *α-Rank* descriptive framework, the method proposed here for computing strategy profile rankings in dynamic games is as follows:

(1) Identify players' styles of play.
(2) Define the strategies of the empirical game based on those styles.
(3) Train policies realizing the defined strategies.
(4) Run game simulations to create the empirical payoff matrix P.
(5) Apply *α-Rank* to define $\mathcal{R}$ and $\mathcal{D}$:
   (a) Calculate the Markov transition matrix C.
   (b) Find the unique stationary distribution π.
   (c) Rank joint strategies by ordering the masses of π.
   (d) Describe the rankings through the response graph.
   (e) Study the effect of different α values on π.

[2]The system $\pi(C - \mathbb{I}) = 0$ by itself does not have a unique solution, as there are infinitely many vectors π that satisfy it. To get a unique solution $\pi = (\pi_1, \pi_2, ..., \pi_{|(SP)|})$, it must hold that $\sum_i \pi_i = 1$.

5 EXPERIMENTS AND RESULTS

In this section, we present the experiments and discuss the results obtained from applying the proposed methodology to the *Graph Coloring Game* using the configuration shown in Figure 1. Appendix *C* provides results with additional grid configurations.

5.1 The Graph Coloring Problem

The well-known *Graph Coloring Problem* (GCP) involves assigning colors to vertices in a graph such that no two adjacent vertices share the same color, and using the minimum number of colors, also known as the chromatic number [25].

In this study, we shift our focus from finding the chromatic number across graph configurations to solving the multi-agent problem of assigning colors to vertices of a dynamic graph with respect to the constraints: In doing so, we define the graph coloring problem as a dynamic game that allows us to study the evolutionary dynamics in multi-agent interactions. This problem setting abstracts settings where agents need to abide to dynamically evolving constraints while acting jointly. For instance, in traffic, any agent acts to abide to constraints, and these actions, given also the dynamics of the environment, result into new emerging constraints to which agents must adhere to, and so on. Through this problem, we aim to demonstrate how we can gain insights into the effectiveness of playing the dynamic game when individual styles of play are combined.

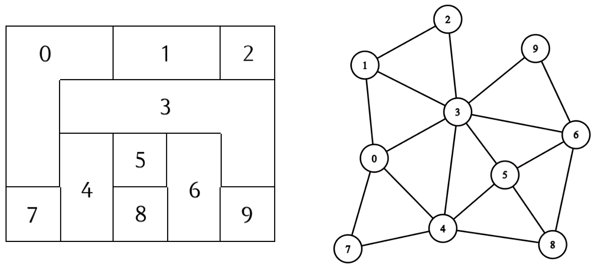


Figure 1: A snapshot of the game environment in grid and graph forms.

We consider the underlying graph coloring dynamic game to be a two-player game executed in rounds. The graph corresponds to a grid comprising blocks of cells: A block comprises one or more merged cells. Each vertex of the graph corresponds to a block, and the adjacency relation between blocks specifies the edges in the graph. At the beginning of the game, the grid is initialized with a random number of rows and columns ($n \times m$). In our experimental setup we assume a 4×5 grid. The environment is initialized by randomly combining cells to create the blocks. The resulting configuration remains the same throughout the entire game. A snapshot of such a configuration with 10 blocks is the one shown in Figure 1, together with the corresponding graph. Merging cells is important as it allows for complex neighboring relationships to be defined, expanding beyond the standard constraints between adjacent blocks.

Blocks are either (a) colored by the agents, (b) white (free to be colored), or (c) hidden (their colors cannot be observed and they cannot be recolored by the agents). Let B be the set of blocks corresponding to graph vertices and CR be the set of possible colors that an agent can use for coloring blocks in B. The game unfolds over multiple rounds in which agents choose their actions simultaneously. At the beginning of each round, the environment reveals the color of some of the hidden blocks, if any. The number of blocks that get unhidden is random, which implies that the state of the graph is influenced not only by the agents' actions but also by the environment. We therefore consider the game to be stochastic. When all blocks in B are uncovered and colored, the game ends.

The set of agents' actions A is defined to be the Cartesian product of the set of blocks B and the set of the available colors CR:

$$A = B \times CR = \{(b, c) \mid b \in B,\ c \in CR\} \tag{5}$$

To specify states, let CR^* include the elements of CR, and two additional elements representing hidden and white blocks: $CR^* = CR \cup \{\text{hidden, white}\}$. A state s is as follows:

$$\begin{aligned} s = \{(b_i, c_i), i = 1, \ldots, |B|\}, \\ s.t. \forall b \in B,\ \exists \text{ a unique } c \in CR^*, \text{ with } (b, c) \in s \end{aligned}$$

Regarding the reward function, it is a sum of gains, penalties, sanctions, delays and adopted preferences. Given that actions are represented as vectors of shape $(b, c) \in B \times CR$, an agent receives a gain point (+1) for each neighbor of b that has a different color than the chosen color c. On the contrary, an agent receives a penalty point (-2) for each neighbor that shares the same color c. Sanction is a big negative reward (-10) that an agent receives when it attempts to color a hidden block or a block that has already been colored. Delay (-1) is a small negative reward that both agents receive when they try to color the same block b, causing a brief pause in the game to determine which agent will eventually color b. Last but not least, there is the preference-adoption reward, which agents receive regardless of whether their action is good, bad, or forbidden. This reward helps agents to be trained so as to adhere to specific preferences, or what we call *styles of play*. We will elaborate shortly on these in the following section.

5.2 Defining the Empirical Game

Transforming the underlying dynamic graph-coloring game into its empirical form involves two key steps: (1) identifying agents' strategies and (2) constructing the empirical game payoff matrix.

5.2.1 Agents' Strategies. Agents' strategies define distinct styles of play, usually revealed by preferences in playing the game. In our experiments, we specify different styles across three main dimensions: color tone (preference for which colors to use), block difficulty (preference for the types of blocks to choose), and coloring approach (preference for the number of colors to use), as shown in Table 2. Through the combination of preferences in each of these dimensions, a style can range from complete indifference, where none of the dimensions hold any influence (denoted by "I"), to specific preferences in all dimensions.

Policies corresponding to specific strategies are represented using convolutional neural networks. To train these policies, we assign specific values in the three dimensions of the game's *preference* reward. These values, range from -1 to 1, where 1 indicates a strong preference for a particular dimension. For example, a value of 0.7 for warm colors indicates a relatively high preference for warm tones. In our experimental setting, we define 11 distinct styles: C, W, E, M, L, A, AE, CA, LE, WL, and I, also with combinations of the preference values specified in Table 2. Assuming no inherent bias among the empirical game players, we allow populations to sample from the same list of strategies.

Preference Dimension	*Value*
Color Tone	warm (W) vs. cool (C)
Block Coloring Difficulty	small (L) vs. large (A)
Coloring Approach	minimalistic (M) vs. extravagant (E)

Table 2: Dimensions specifying agents' strategies

5.2.2 *Training the agents.* All policy models share the same underlying architecture and training setup. Although hyperparameter tuning is typically recommended, it does not make much difference in this case, as these models are relatively easy to optimize when trained in small settings. Regarding the convolutional neural network architecture, it consists of four convolutional layers, each defined with a kernel size of 3, stride of 1, and padding of 1, meant to extract spatial features from the input. The input tensor has dimensions 10×12, where $|B| = 10$ represents the number of blocks in the state and $|CR^*| = 12$ represents the number of possible colors a block can have. Each block is encoded using one-hot encoding, meaning that each color is represented as a binary vector of length 12. As shown in Figure 2, the network processes the input through four convolutional layers with respective output sizes $10 \times 12 \times 32$, $10 \times 12 \times 64$, $10 \times 12 \times 128$, and $10 \times 12 \times 256$. The final convolutional layer output is flattened to a vector of size 30720. This representation is passed through two fully connected layers: the first reduces the size to 512, and the second produces the final output, which corresponds to a matrix of size $|CR| \times |B| = 10 \times 10$ encoding the action space. Although the architecture seems to be excessive, it aims at efficient policy training and generalization, by extracting meaningful relationships among state components (this is justified in Appendix *D*).

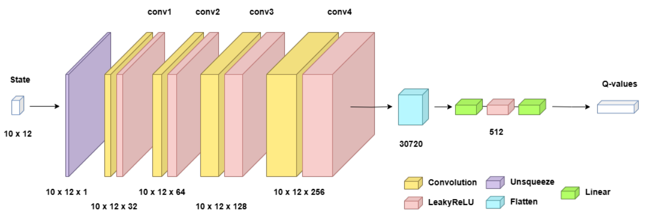


Figure 2: Convolution Policy Network Architecture

Policy models are trained individually (without co-players) in the underlying game using the deep Q-learning reinforcement learning algorithm specified in Algorithm 1. We set γ to 0.7. To optimize the model parameters, we use the smooth L1 loss function with β=1.0 and the Adam optimizer with a learning rate of 5e-4 and weight decay of 1e-5 to prevent over-fitting. To further enhance the learning process, we incorporate experience replay, with a memory that stores up to 10 million experiences [11]. A target network alongside the main policy network, is being used according to the Double-DQN approach [24]. To update the target network we apply a soft update with a factor τ=5e-3. This gradually brings the target network closer to the policy network, balancing learning speed and stability. With a batch size of 64, we train the models for 10000 episodes. All trained agents manage to learn a policy that successfully abides to constraints.

Algorithm 1: Double Deep Q-Learning with Experience Replay

```
1:  $Q_\theta, Q_{\theta'} \leftarrow Q_\theta, M$        ▷ Initialize policy/target nets & memory
2:  for episode do
3:      $s \leftarrow s_0$
4:      for step do
5:          $a \leftarrow \text{argmax}_a Q_\theta(s)$        ▷ Select $\epsilon$-greedy action
6:          $(s, a, r, s') \in M$        ▷ Store experience
7:          if $|M| >$ batch size then
8:              for each $(s, a, r, s')$ in $M$ do        ▷ Sample memory
9:                  $y \leftarrow r + \gamma \max_{a'} Q_{\theta'}(s')$
10:                 $L \leftarrow \text{Loss}(Q_\theta(s), y)$
11:                 $\theta \leftarrow \theta - \alpha \nabla_\theta L$
12:             end for
13:         end if
14:         $Q_{\theta'} \leftarrow \tau Q_\theta + (1 - \tau) Q_{\theta'}$        ▷ Soft update
15:         $s \leftarrow s'$
16:     end for
17: end for
```

5.2.3 *Empirical Game Payoff Matrix.* The empirical payoff matrix is generated by simulating each strategy profile over multiple games. Payoffs represent how well different styles of play perform jointly, according to the game's rules. Here we must note that even the most incompatible pairs of styles violate few constraints, due to the inherent differences between agents' preferences. For instance, the profile (W,C) of compatible strategies scores 0.05% of violations, while the profile of incompatible strategies (C,C) scores 23% violations.

The values in the payoff matrix are computed in terms of the delay and the quality of the solution according to the game's constraints (gain, penalty, and sanction), excluding preferences. This ensures a common ground for distinct strategies, evaluating solutions solely based on the game's rules. For each pair of strategies, we simulate the game over 5,000 repeats and calculate the average payoff for each strategy. These values are then organized into the payoff matrix, which is provided in Appendix A. From this matrix, we observe that (L, WL) and its symmetric counterpart (WL, L) both with payoffs of (3.15, 3.21) and (3.21, 3.15) respectively, are the only Nash equilibria. It is important to note here that these equilibria prescribe agents' strategies, given that they do play the game with rational co-players, but they do not capture the overall dynamics of the game, considering the long-term effects of agents' interactions.

5.3 Evaluation and Ranking

Given the payoff matrix derived from the empirical analysis, we apply the α-*Rank* method to evaluate the performance of strategy profiles over time in terms of the MCC solution concept. Specifically, we ran the method 1000 times, using values of α within the range $[0.1, 10]$ with step=0.01, while assuming populations of size $m = 100$. We provide as input the strategies defined in Section 5.2.1 and the empirical game payoff matrix. We focus on the rankings of the top 6 strategy profiles, to identify the stronger ones across different values of α.

As we observe from the rankings in Table 3, the strategy profile that prevails in the long run is (WL, CA); this is the primary component of the MCC. Although the table was derived using an α value of 2, the rankings remain consistent even when α is set to 10. We choose $\alpha = 2$ over $\alpha = 10$, to display the rankings of lower-performing strategy profiles, which would otherwise drop to zero. First, it is worth mentioning that the Nash equilibria (L, WL) and (WL, L) don't appear among the top-ranked strategy profiles. This is because MCC components are defined based on how well strategies perform when interacting with other strategies, based on long-term agents interactions. The individual strategies within the Nash equilibrium profile, either WL or L, may not result in favorable interactions with other strategies. As a result, the profile (WL, L) is ranked lower than others.

To further support our observations regarding the misalignment between the two solution concepts, let's examine why (CA, WL) is part of the MCCs, while (L, WL), the Nash equilibrium, is not. A closer look at the payoff matrix in Table **??** reveals that L appears to be the worst-performing strategy for the row player, with an average payoff of 3.13. In this case, being in the Nash equilibrium means the player is stuck with a strategy that gives low rewards, making it the best among other options, rather than a strong choice. If it happens to play this strategy, it would expect its rational opponent to play WL. Strategy CA on the other hand, is the best-performing strategy for the row player, with an average payoff of 3.18. Combined with WL, which is the best performing strategy for the column player, with an average payoff of 3.18, make (CA, WL) to be the top ranked strategy profile in the ranking Table 3.

Profile	Rank	Score
(WL, CA)	1	0.42
(W, CA)	2	0.13
(M, CA)	3	0.12
(CA, M)	4	0.08
(CA, W)	5	0.08
(CA, LE)	6	0.01

Table 3: Strategy profiles' rankings for $\alpha = 2$

Rankings within the MCC are also very intuitive. For example, strategies that prefer different color tones, such as (WL, CA) or (W, CA), tend to result into fewer conflicts since, they naturally avoid selecting the same colors. Similarly, strategies that prefer different blocks based on their difficulty, such as (WL, CA) or (CA, LE), tend to provide solutions with minimal delay, as they naturally avoid coloring the same blocks. Notably, profiles with mixed preferences across these dimensions demonstrate the most promising performance, which explains why (WL, CA), as such a profile, is a key component of the MCC. However, not all profile rankings can be easily explained through the game's rules alone; the expected influence of certain strategies on the quality of the solutions remains ambiguous. For example, profiles with strategies like M and E are more difficult to analyze.

The response graph provides a visualization to interpret the α-*Rank* results. This graph illustrates the MCC, using the strategy profiles' masses from the stationary distribution, π, along with the fixation probability function ρ provided by α-*Rank*. Figure 3 shows the response graph for $\alpha = 6.4$. We consider it to be part of the descriptive framework $\mathcal{D}$, as it offers insights into how rankings were derived. Additional graphs for $\alpha = 0.4$, 1.3, and 1.9 are available in Appendix B.

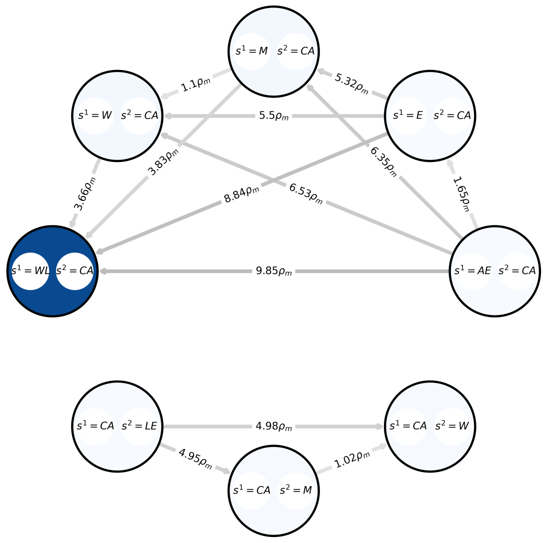
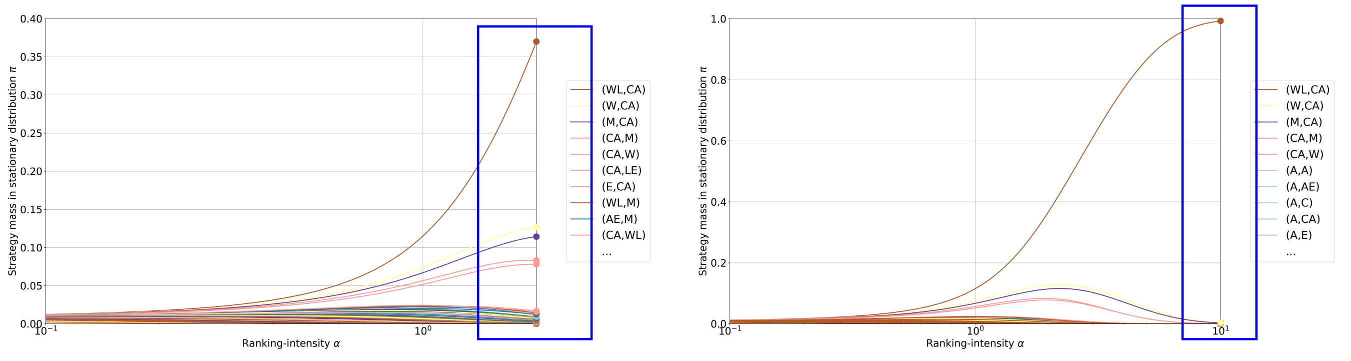


Figure 3: Response graph for $\alpha = 6.4$.

The response graph describes the dynamics of the strategy profiles in the empirical game. One prominent feature is the primary component of the MCC: the profile (WL, CA). This profile, indicated by a dark blue color, has multiple graph edges leading to it, while none from it, indicating that strategies in this profile are non-transient. This is further supported by the large fixation probabilities along the edges. A particularly prominent example is the cluster (CA, LE)-(CA, M)-(CA, W), which consists of three strongly connected profiles, indicating that once a player adopts one of these profiles, they will likely remain within their cluster. These components reflect stable regions in the game's strategy dynamics, where transitions between profiles become locked into a cycle.

To further investigate the effect of α on profile dominance, we plotted the stationary distribution π across all α values used in the experiments, for the top-performing strategy profiles (see Figure 4). This visualization–also part of $\mathcal{D}$–helps us understand how the stationary distribution changes as the selection intensity increases. The x-axis represents the different α values, ranging from 0.1 to



(a) Mass across $\alpha \in [0.1, 3]$ **(b) Mass across $\alpha \in [0.1, 10]$**

Figure 4: Effect of ranking intensity α on strategy profile mass in the stationary distribution π. The order of profiles in the legend matches the mass for maximal α.

3 in Figure 4a, and from 0.1 to 10 in Figure 4b, while the y-axis in both figures shows the mass of each strategy profile in the stationary distribution π. As α increases, the distribution converges, and the selection process stabilizes. The final mass distributions are highlighted in boxed regions. The legend at the right displays the top-performing joint strategies, with the stronger ones at the top.

We plot two such graphs to observe how the mass of strategy profiles is distributed in the MCCs across different α values. In the stationary distribution resulting from a bigger α, the dominant strategy profile (WL, CA) in the MCC achieves a mass of 1, with all other profiles dropping to 0. This is clearly illustrated in the second plot (see Figure 4b). However, regarding the mass distribution for a smaller range of α, depicted n the first plot, the game has not yet converged to the final MCC.

6 RELATED WORK

Evaluation and ranking of learned multi-agent strategies is mainly based on game theoretic concepts, while computational social choice has been proposed, as well [9]. A predominant approach is the Elo rating system used to evaluate and rank agents that learn through reinforcement learning ([7], [18], [19], [12]). Elo estimates the probability an agent to win another agent. Although it was designed specifically to rank players in the two-player, symmetric constant-sum game, it has been widely applied to other domains. Recently, it has been used for the evaluation of large language models [28]. However, Elo cannot model intransitive relationships [1], as those in the Rock-Paper-Scissors game, in real-world games [4], and in our game coloring setting. In addition, incorrectness issues have been reported in transitive settings [2]. Nash-based evaluation methods, such as Nash Averaging, can be applied in two-player, zero-sum settings ([1] [21]), but these are not more generally applicable as the Nash equilibrium is intractable to compute and select [5].

Here, our focus is on (many) agents' long-term interactions in general-sum dynamic settings: Agents need to rank their policy profiles to decide their non-transient individual policy profile, accounting for long-term interactions and payoffs.

Empirical Game Theory Analysis (EGTA), deploying empirical or meta-games [15] [20], [8], [26], can be used to evaluate learning agents that interact in large-scale multiagent systems [13], [22] [23]. In our case, aiming at strategy profile rankings, we define the empirical game strategies based on agents' styles of play and train policies realizing these strategies. Then the empirical payoff matrix is estimated by means of game-playing simulations, and is used by the α-Rank method to compute strategy profile rankings. α-Rank applies to many-player, general-sum games [13].

7 CONCLUSION

In this study, we developed a methodology for identifying strong joint-strategies in dynamic multi-agent games, accounting for stability and performance, using the *α-Rank* evolutionary algorithm. This methodology is applied to a stochastic version of the *Graph Coloring Problem*, where players collaborate to color a graph while ensuring that neighboring vertices are assigned different colors. According to the methodology, we first transformed the game into its empirical form by defining styles of play. We then designed and trained Deep Q-Learning policy models that realize these styles of play in the underlying game, and run simulations to generate the empirical payoff matrix. Applying *α-Rank* to this matrix results in a unique stationary distribution over strategy profiles, which defines the empirical game's MCC. *α-Rank* not only helped us identify stable strategy profiles resistant to changes, but also provided a descriptive framework for understanding why certain profiles prevail in the long run, based on the underlying dynamics of the game. Through this approach, we successfully described a concise methodology for evaluating and ranking agents' joint policies, considering their long-term interactions in dynamic settings, while also explaining how strategy profiles are defined within the MCC.

Future work involves (a) applying the methodology in more complex and large-scale settings, accounting for strategy profiles of multiple stakeholders that may collaborate and/or compete, (b) using machine learning methods to identify different styles of play from demonstrations and specifying the empirical game, (c) exploring models able to adapt their strategies based on observed behaviors of co-players effectively, and (d) applying the methodology into real-world settings where agents need to align with human preferences in dynamic settings.

REFERENCES

[1] David Balduzzi, Karl Tuyls, Julien Pérolat, and Thore Graepel. 2018. Re-evaluating evaluation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 3272–3283. https://proceedings.neurips.cc/paper/2018/hash/cdf1035c34ec380218a8cc9a43d438f9-Abstract.html

[2] Quentin Bertrand, Wojciech Marian Czarnecki, and Gauthier Gidel. 2023. On the Limitations of the Elo, Real-World Games are Transitive, not Additive. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 206)*, Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (Eds.). PMLR, 2905–2921. https://proceedings.mlr.press/v206/bertrand23a.html

[3] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. 2015. Evolutionary dynamics of multi-agent learning: a survey. *J. Artif. Int. Res.* 53, 1 (May 2015), 659–697.

[4] Wojciech M. Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. 2020. Real World Games Look Like Spinning Tops. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 17443–17454. https://proceedings.neurips.cc/paper_files/paper/2020/file/ca172e964907a97d5ebd876bfdd4adbd-Paper.pdf

[5] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. 2006. The complexity of computing a Nash equilibrium. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing* (Seattle, WA, USA) *(STOC '06)*. Association for Computing Machinery, New York, NY, USA, 71–78. https://doi.org/10.1145/1132516.1132527

[6] Omid E. David, H. Jaap van den Herik, Moshe Koppel, and Nathan S. Netanyahu. 2014. Genetic Algorithms for Evolving Computer Chess Programs. *IEEE Transactions on Evolutionary Computation* 18, 5 (Oct. 2014), 779–789. https://doi.org/10.1109/tevc.2013.2285111

[7] A.E. Elo. 1978. *The Rating of Chessplayers, Past and Present.* Arco Pub. https://books.google.gr/books?id=8pMnAQAAMAAJ

[8] William E Walsh Rajarshi Das Gerald and Tesauro Jeffrey O Kephart. 2002. Analyzing Complex Strategic Interactions in Multi-Agent Systems. (2002).

[9] Marc Lanctot, Kate Larson, Yoram Bachrach, Luke Marris, Zun Li, Avishkar Bhoopchand, Thomas Anthony, Brian Tanner, and Anna Koop. 2023. Evaluating Agents using Social Choice Theory. *arXiv preprint arXiv:2312.03121* (2023).

[10] Michael Levet. 2016. Game Theory : Normal Form Games. https://api.semanticscholar.org/CorpusID:131771375

[11] Long-Ji Lin. 1992. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Mach. Learn.* 8, 3–4 (may 1992), 293–321. https://doi.org/10.1007/BF00992699

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[13] Shayegan Omidshafiei, Christos Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M Czarnecki, Marc Lanctot, Julien Perolat, and Remi Munos. 2019. α-rank: Multi-agent evaluation by evolution. *Scientific reports* 9, 1 (2019), 9937.

[14] Sheryl Paul and Jyotirmoy V. Deshmukh. 2022. Multi Agent Path Finding using Evolutionary Game Theory. arXiv:2212.02010 [cs.MA] https://arxiv.org/abs/2212.02010

[15] Steve Phelps, Simon Parsons, and Peter McBurney. 2004. An evolutionary game-theoretic comparison of two double-auction market designs. In *Proceedings of the 6th AAMAS International Conference on Agent-Mediated Electronic Commerce: Theories for and Engineering of Distributed Mechanisms and Systems* (New York, NY) *(AAMAS'04)*. Springer-Verlag, Berlin, Heidelberg, 101–114. https://doi.org/10.1007/11575726_8

[16] Lloyd S. Shapley. 1953. Stochastic Games*. *Proceedings of the National Academy of Sciences* 39 (1953), 1095 – 1100. https://api.semanticscholar.org/CorpusID:263414073

[17] Yoav Shoham and Kevin Leyton-Brown. 2008. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

[18] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (01 2016), 484–489. https://doi.org/10.1038/nature16961

[19] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144. https://doi.org/10.1126/science.aar6404 arXiv:https://www.science.org/doi/pdf/10.1126/science.aar6404

[20] Karl Tuyls and Simon Parsons. 2007. What evolutionary game theory tells us about multiagent learning. *Artificial Intelligence* 171, 7 (2007), 406–416.

[21] Karl Tuyls, Julien Perolat, Marc Lanctot, Joel Z. Leibo, and Thore Graepel. 2018. A Generalised Method for Empirical Game Theoretic Analysis. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (Stockholm, Sweden) *(AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 77–85.

[22] Karl Tuyls, Julien Perolat, Marc Lanctot, Joel Z Leibo, and Thore Graepel. 2018. A generalised method for empirical game theoretic analysis. *arXiv preprint arXiv:1803.06376* (2018).

[23] Karl Tuyls, Julien Pérolat, Marc Lanctot, Georg Ostrovski, Rahul Savani, Joel Z Leibo, Toby Ord, Thore Graepel, and Shane Legg. 2018. Symmetric decomposition of asymmetric games. *Scientific reports* 8, 1 (2018), 1015.

[24] Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep Reinforcement Learning with Double Q-learning. arXiv:1509.06461 [cs.LG]

[25] George Watkins, Giovanni Montana, and Juergen Branke. 2023. Generating a Graph Colouring Heuristic with Deep Q-Learning and Graph Neural Networks. arXiv:2304.04051 [cs.LG]

[26] Michael P Wellman. 2006. Methods for empirical game-theoretic analysis. In *AAAI*, Vol. 980. 1552–1556.

[27] Michael P. Wellman, Karl Tuyls, and Amy Greenwald. 2024. Empirical Game-Theoretic Analysis: A Survey. arXiv:2403.04018 [cs.GT] https://arxiv.org/abs/2403.04018

[28] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and Chatbot Arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 2020, 29 pages.