**2020 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 15-16, Ponta Delgada, Portugal**

Competitive Deep Reinforcement Learning over a Pokémon Battling Simulator

David Simões
Institute of Electronics and Informatics Engineering of Aveiro
University of Aveiro
Aveiro, Portugal
david.simoes@ua.pt

Simão Reis
Artificial Intelligence and Computer Science Laboratory
University of Porto
Porto, Portugal
simao.reis@outlook.pt

Nuno Lau
Institute of Electronics and Informatics Engineering of Aveiro
University of Aveiro
Aveiro, Portugal
nunolau@ua.pt

Luís Paulo Reis
Artificial Intelligence and Computer Science Laboratory
University of Porto
Porto, Portugal
lpreis@fe.up.pt

Abstract—**Pokémon is one of the most popular video games in the world, and recent interest has appeared in Pokémon battling as a testbed for AI challenges. This is due to Pokémon battling showing interesting properties which contrast with current AI challenges over other video games. To this end, we implement a Pokémon Battle Environment, which preserves many of the core elements of Pokémon battling, and allows researchers to test isolated learning objectives. Our approach focuses on type advantage in Pokémon battles and on the advantages of delayed rewards through switching, which is considered core strategies for any Pokémon battle. As a competitive multi-agent environment, it has a partially-observable, high-dimensional, and continuous state-space, adheres to the Gym *de facto* standard reinforcement learning interface, and is performance-oriented, achieving thousands of interactions per second in commodity hardware. We determine whether deep competitive reinforcement learning algorithms, WPLθ and GIGAθ, can learn successful policies in this environment. Both converge to rational and effective strategies, and GIGAθ shows faster convergence, obtaining a 100% win-rate in a disadvantageous test scenario.**

Index Terms—**Deep Learning, Reinforcement Learning, Competitive Games, Multi-Agent Systems**

I. Introduction

Pokémon [12] is the largest entertainment franchise in the world, having at its core a role-playing video game series. In Pokémon games, the player travels through a region inhabited by Pokémon specimens and can capture, collect, and train them to finally participate in battle competitions. In Pokémon battles, the trainer's goal is to defeat the opponent's Pokémon team before they defeat his. The Pokémon core videogame series re-iterates over the years with new sequel releases, each one adding new mechanics to the Pokémon battles, but with unchanged core mechanics and goals. Some of the most notable changes of more recent Pokémon generations are the addition of passive abilities, held items that trigger under a certain game condition, restructuring of damage calculation or even the addition of new Pokémon types among others.

This work is supported by: Portuguese Foundation for Science and Technology (FCT) under grant PD/BD/113963/2015 and SFRH/BD/129445/2017; Institute of Electronics and Informatics Engineering of Aveiro – IEETA (UIDB/00127/2020); Artificial Intelligence and Computer Science Laboratory – LIACC (UIDB/00027/2020).

Games have been used as test-beds for artificial intelligence algorithms because they provide a controlled environment with widely different sets of rules and levels of complexity. They allow researchers to demonstrate properties of their algorithms in single-agent environments [24] (where a single agent competes to successfully finish the game) and in multi-agent systems (where a team of agents must coordinate to achieve a goal [27], [30], [29], or a group of enemies must compete for supremacy [32], [11], [26]). Pokémon battling falls under the latter category, a competitive 1v1 environment, where each agent (or trainer) controls a Pokémon, turn by turn, to attempt to defeat the enemy.

In a Pokémon battle, a player competes with a team of up to six Pokémon, and each Pokémon possesses a set of statistics such as hit points (HP), attack, defense, speed modifiers and up to four moves (or attacks). In 1v1 battles, each player controls one active Pokémon fighting against the opponent's active Pokémon and holds the remaining on the bench. Each turn, the player may select one of four moves for his active Pokémon to attack the opponent's active Pokémon, or the trainer may switch the active Pokémon with a benched one. When a Pokémon's HP depletes, it becomes unusable for the remainder of the battle and is forcibly switched out.

A core component of Pokémon is typing, as each Pokémon and move possesses a type, and types have different effectiveness against different types. A Fire-type move is super effective against Grass-type Pokémon, and Water-type Pokémon resist Fire-type moves. The effectiveness of move typing is translated into a modifier chart (asymmetrical matrix) where the base multiplier is 1x, weakness has a 2x multiplier, resistance has a 0.5x multiplier and immunity has a 0x multiplier.

The game features several properties that make it a challenging environment for machine learning algorithms. It is partially-observable since each trainer only knows information about its team and some visible stats regarding the opponent's active Pokémon. Unlike chess or Go, the environment is stochastic, and attacks have a chance to hit and do damage based on an interval formula. It is a multi-agent system, and as the trainer learns and optimizes his strategies, the opponent is learning and adapting as well, a situation known as the moving target problem [4]. The reward scheme is zero-sum [6], since the players' rewards are symmetrical, and the player's only way to win is through the opponent's defeat. The state-space is continuous and high-dimensional, with six Pokémon for each trainer, each with one or two of seventeen different Pokémon types, four attacks per Pokémon, each with its typing as well, hit points and statistics for each Pokémon, and power and accuracy values for each attack. To achieve successful policies in a reasonable amount of time, algorithms must generalize to new unseen states, since exploring the entire state-space is intractable. This set of properties requires algorithms that are targeted at complex competitive games, and capable of coping with partially-observable, continuous and high-dimensional state-spaces. The environment is ideal to test and demonstrate the properties of such algorithms.

The main contributions of this work are two-fold: (i) the development of a machine-learning oriented environment that replicates the core gameplay mechanics of Pokémon battles; (ii) an empirical demonstration of the feasibility of multi-agent deep learning algorithms, never tested in literature in this new type of scenario with interesting properties in terms of an AI challenge. These can later be applied in competitive robotic environments, like Robocup [18].

The remainder of this paper is organized as follows. In Section II, we discuss related work with regards to Pokémon simulators and competitive environments for AI competitions, as well as a proposed Showdown AI Competition, a Pokémon battle competition for AI agents. We also review some state of the art solutions for machine learning in competitive multi-agent settings in Section III, which we use later on. Our proposed environment is described in section IV, and our experimental setup and results are shown in Section V. Finally, Section VI lists our conclusions.

II. Related Work

Multiple Pokémon environments have been developed over the years, both officially released [12], or fan-made [33], [9]. Neither official games nor fan-made battle simulators provide adequate application programming interfaces (API) for machine learning algorithms and are instead focused on human user-experience. To the best of our knowledge, proper API is only available for information mining about the games [10], [14], such as Pokémon lists or maps.

Closer applications of artificial intelligence to Pokémon Battling can be found in the work of Alfonso [2], where hard-coded agents battle each other using official Pokémon statistics. A brief analysis of this work demonstrates that agents are sub-optimal and do not perform tactical long-term decisions, such as switching for other Pokémon in the party when a type disadvantage is presented (e.g., the trainer has an active fire Pokémon and an electric Pokémon in the party, but does not switch them when facing a water Pokémon, which is effective against fire and vulnerable to electric attacks). Kalose et al. [16] focus on Reinforcement Learning and use a single Q-learning [31] agent and show that they can achieve a win-rate over 65% against random opponents, or 90% using a Minimax-Q agent [20]. However, the authors developed a deterministic simulator with a limited amount of Pokémon, types, and attacks, using a discretized-state space that only includes the active Pokémon's hit points, types, and attacks. Approaches based on supervised learning can, on the other hand, be used to predict the likely outcome of Pokémon match-ups [7].

A. Showdown AI Competition

Given the rising interest of machine learning research in the Pokémon Battling field, an AI competition [19] has recently been proposed for the IEEE Conference on Games, known as Showdown AI Competition. The authors state that most current competitions are based on real-time video games with perfect information and that there is a need to explore other types of games with competitive nature.

The authors identify eight main properties that contrasts Pokémon Battling with other games:

- Branching Factor - with (on average) nine possible actions per turn (there are four possible moves and five possible switches), planning multiple steps is computationally heavy. This of course in the assumption that in competitive Pokémon battles the use of items are not allowed;
- Infinite Looping - both agents may choose to switch endlessly without causing any damage to the opponent Pokémon. Although this rarely happens in human games, agents may not identify this cycle to break it;
- Turn Atomicity - although Pokémon is a turn-based game, choices are made simultaneously, and agents only observe both moves after selection;
- Categorical Dimensions - although in a clean state, HP and statistics are sufficient to evaluate the actions' reward, over-time conditions like *burned* or *paralyzed*, or other field effects like *sandstorm* or *stealth rock* are hard to quantify (delayed rewards);
- Stochasticity - moves' damage calculation have random parameters, and may miss and do zero damage;
- Hidden Information - this environment is partially-observable at two levels. The opponent's active Pokémon's move set, statistics and abilities (although this can be minimized with domain knowledge), and at the team level, the unawareness of the opponent's

party makes it harder to plan and predict best long-term strategy;
- Deception - caused by a single ability, where a Pokémon appears in battle disguised as another Pokémon from the opponent's party;
- Computational Cost - a Pokémon simulator, due to having low graphical requirements, can probably outperform many other video-game simulators.

These properties motivate the existence of a Pokémon battle simulator that is compliant with standard machine learning interfaces.

The showdown AI competition also identifies two main categories of domain knowledge, which can help the complexity of the large state space. The first is Pokémon typing, which helps to estimate the damage and move value. The other is archetypes, wherein formats where players are allowed to build their teams, knowing synergies between moves, or Pokémons themselves can help to predict the opponent team, but this consideration is out of scope of this work.

III. Competitive Reinforcement Learning

Competitive games often require agents to follow stochastic strategies, also known as mixed policies [6], where each action is played with some probability. Mixed-policy algorithms that consider the non-stationarity of the environment without having unrealistic assumptions or expectations about other agents are ideal for such environments. They are also the most well-studied algorithms in the literature, the most common and general, and they are often computationally inexpensive. The goal for most is to reach a Nash equilibrium [25], a strategy that cannot be improved and that the opponent cannot take advantage of. The majority of algorithms are derived from Q-learning [15] and keep track of both Q-values and of a probability distribution in each state. This probability may tend to a pure strategy, where the algorithms become the original greedy Q-learning. They often update their Q-values in the same manner as Q-learning and introduce different ways of calculating the policy π. GIGA-WoLF [3], [28] relies on the *Win or Learn Fast* (WoLF) principle, where different learning rates are used when the agent is winning or losing. WPL [1] instead uses a variable learning rate, but has no formal analysis and proof of convergence.

Both algorithms have been extended to the deep learning paradigm as GIGAθ and WPLθ [28], based on Asynchronous Q-Learning [21], which is a faster-distributed version of Deep Q-Learning (DQN) [23]. Asynchronous methods are more adequate for multi-agent learning because they rely on parallel threads (known as workers) to gather samples to update their neural networks. DQN, on the other hand, uses a single worker and accumulates samples in a memory buffer. While this makes it more sample efficient, when used in a multi-agent setting, it causes agents to use outdated samples and adapt to opponent strategies that have since evolved. GIGAθ and WPLθ are specifically targeted at competitive environments and have been shown to allow players to converge to the Nash Equilibrium strategy in simple games [28].

A. Generalized Infinitesimal Gradient Ascent

Generalized Infinitesimal Gradient Ascent using the WoLF principle (GIGA-WoLF) [3] keeps track of two gradient updated strategies, one of which is updated faster than the other. *Regret* measures how much worse a policy performs compared to the best static strategy, and GIGA-WoLF exhibits both no-regret and convergence properties. We refer the reader to Simões et al. [28] for further information. Using GIGA-WoLF's equations, an asynchronous deep version of GIGA-WoLF is described in Algorithm 1 as GIGAθ. Two policy networks with weights θ_π and $\theta_{\hat{\pi}}$ are used, but only a single policy update rate δ is required.

```
Input: Globally, target network update period $\tau$, on-line and target Q-networks,
    on-line and target policy networks, on-line and target average policy network
    weights, exploration rate $\epsilon$, and maximum iterations $T_{\max}$. Locally, on-line Q,
    policy and average policy networks.
1: for iteration $T \leftarrow 0, T_{max}$ do
2:     Synchronize local on-line networks as copies of global on-line networks
3:     Sample initial state
4:     repeat
5:         Execute random action with probability $\epsilon$, otherwise execute action
           from policy network
6:         Sample new state and reward
7:         Compute Q-network targets with target Q-network
8:         Compute policy networks' targets with GIGA-WoLF's update equations
9:         Compute loss of local on-line Q, policy, and average policy networks
10:        Accumulate gradients by minimizing the loss
11:    until terminal state
12:    Update global on-line networks weights with the accumulated gradients of
       local on-line networks
13:    Synchronize global target networks as copies of global on-line networks
       every $\tau$ time-steps
14: end for
Output: A converged Q-network to approximate the value function as $Q(s, a, \theta)$,
    and a converged policy network to approximate the policy function as
    $\pi(s, a, \theta_\pi)$.
```

Algorithm 1: Pseudo-code for a worker thread running GIGAθ using ϵ-greedy exploration.

B. Weighted Policy Learner

Weighted Policy Learner (WPL) [1] has a variable learning rate and allows the agent to move towards the equilibrium strategy faster than moving away from it. Despite not having a formal proof of convergence due to the non-linear nature of WPL's dynamics, the authors numerically solve WPL's dynamics differential equations and show that it features continuous non-linear dynamics, while experimentally demonstrating it converges in several more complex games. We refer the reader to Simões et al. [28] for further information. Using WPL's equations, an asynchronous deep version of WPL is described in Algorithm 2 as WPLθ. Unlike the previous algorithm, a single policy network θ_π is used in conjunction with a Q-network.

IV. Pokémon Battle Environment

On Pokémon Battle Environment (PBE), each team is composed of an active Pokémon, and a benched Pokémon. Players have access to their Pokémon's maximum and current HP, typing, and semi-random type moves. Each move has a power uniformly distributed between 50 and 100 (moves with the same type as their owner benefit from a power

```
Input: Globally, target network update period $\tau$, on-line and target Q-networks,
    on-line and target policy networks, exploration rate $\epsilon$, and maximum iterations
    $T_{\max}$. Locally, on-line Q and policy networks.
1:  for iteration $T \leftarrow 0, T_{max}$ do
2:      Synchronize local on-line networks as copies of global on-line networks
3:      Sample initial state
4:      repeat
5:          Execute random action with probability $\epsilon$, otherwise execute action
            from policy network
6:          Sample new state and reward
7:          Compute Q-network targets with target Q-network
8:          Compute policy networks' targets with WPL's update equations
9:          Compute loss of local on-line Q and policy networks
10:         Accumulate gradients by minimizing the loss
11:     until terminal state
12:     Update global on-line networks weights with the accumulated gradients of
        local on-line networks
13:     Synchronize global target networks as copies of global on-line networks
        every $\tau$ time-steps
14: end for
Output: A converged Q-network to approximate the value function as $Q(s, a, \theta)$,
    and a converged policy network to approximate the policy function as
    $\pi(s, a, \theta_\pi)$.
```

Algorithm 2: Pseudo-code for a worker thread running WPLθ using ϵ-greedy exploration.

boost of 50%). Each Pokémon has at least one move of its type, and three moves of random types that are not effective against the Pokémon and that the Pokémon is not effective against. For example, a Fire Pokémon will not have a Water attack (effective against Fire) or a Grass attack (which Fire is effective against). While some specific Pokémon have attacks with unconventional types, removing these from the PBE leads to more strategic policies learned by agents than if a Pokémon could have fully-random type moves. Agents do not initially know their opponent's moves, only their HP and type. Through the course of a battle, they can learn information about the opponent as it uses its attacks.

Agents have a reward function

$$R = R_d + R_f - R_t,$$

where R_d represents the damage dealt as a fraction of the opponent's maximum HP, $R_f \in 0, 1$ is a bonus if the opponent fainted, and R_t represents the damage taken as a fraction of Pokémon's maximum HP. With the feedback provided each turn, this results in non-sparse rewards. If a Pokémon faints, it automatically switches to the benched Pokémon, and a battle ends when a trainer's Pokémon have all fainted. The order of attacks is random each turn, leading to stochastic state transitions. The switch action, like the official Pokémon games always occur before attacks.

The 5-dimensional action space of each agent corresponds to the four moves from the active Pokémon, and the switch option. If the benched Pokémon has fainted, the switch action wastes a turn (an agent should avoid choosing it in such cases). Agents sample actions simultaneously and observe the state-space, composed of the status (HP and type) of all his Pokémon, and the HP of the opponent's active Pokémon. Additionally, the state space contains the power and type of all four moves of the active Pokémon. With 18 different types, and identifying them through a one-hot vector, this is equivalent to dimensional state space with over 100 dimensions for each agent.

Overall, the most important components of the core gameplay of Pokémon battles are replicated in a controlled environment, which allows researchers to test various learning objectives. An obvious one is a capacity for agents to learn typing combinations, and optimizing the move taken to cause the most damage to the opponent. For example, a Fire move is effective against a Grass Pokémon, but resisted by a Water Pokémon. Another learning goal is the optimization of a long-term strategy in favor of a worse strategy with immediate rewards. The switch action will give the agent a reward $R \leq 0$, but switching to a Pokémon with better typing and move pool often increases the overall reward obtained. These core strategies are sufficient for players to complete official Pokémon games, by training and battling the main enemies in the game.

Comparing the PBE environment with the one from the Showdown AI Competition, in terms of AI challenge, we identify a set of differences and similarities. The branching factor was decreased, there are no long-term conditions or mechanics, and the Deception ability does not exist, all of which create a simpler environment. However, stochasticity is heavily exacerbated with a random set of generated Pokémon and moves, instead of a fixed set of viable combinations and teams. PBE also maintains multiple similarities, since turn atomicity is preserved, infinite looping remains possible, and the environment remains partially-observable. PBE is compliant with Gym [5], a *de facto* standard in the reinforcement learning community. Its source code can be found at https://gitlab.com/DracoStriker/simplified-pokemon-environment.

V. Evaluation

The applicability of reinforcement learning algorithms to PBE is now demonstrated. The GIGAθ and WPLθ algorithms are used, both being competitive mixed-policy deep-learning algorithms that require only local information to converge to Nash equilibrium policies. Algorithms have no prior knowledge about the environment and must learn strategies that include type effectiveness and tactical switches. Tests were conducted in a complete version of PBE, and in a simpler version with only seven different types.

The asynchronous mixed policy tests used 12 concurrent workers, a neural network with three hidden layers of 150 nodes with ELU activations [8], a learning rate $\eta = 10^{-4}$, a policy update rate $\delta = \frac{\eta}{200}$, and a future reward importance discount $\gamma = 0.9$. The ϵ-annealing technique was used, following the scheme used originally in Mnih et al. [22], with each worker annealing the exploration rate from 1 to one of $0.5, 0.1, 0.01$ over 1.3 million episodes. Weights were initialized with the Glorot initializer [13] and optimized with the Adam optimizer [17], using standard parameters.

Both algorithms trained in self-play for two million episodes and were tested in a fixed scenario, where a trained agent in a disadvantageous position played against a random enemy. The learning results are shown in Figure 1, and compared against the hard-coded baseline that followed the guidelines given by two human experts. The hard-coded solution emulates an expert player's decision making, maximizing the damage output of the team, based on the known information about the opponent's team. Both GIGAθ and WPLθ achieve good results in the 7-type PBE, but GIGAθ converges faster and can match the performance of the hard-coded baseline. In the complete version, WPLθ is no longer able to learn adequate policies. On the other hand, GIGAθ has a slower evolution with higher variance, likely due to the increased complexity of the environment, but still matches the performance of the hard-coded solution.

The fixed test scenario puts the trainer at an initial disadvantage, initially favoring the opponent with a Fire/Normal roster, while the trainer has a Grass/Water roster. Because Fire is effective against Grass, and Water against Fire, the allied trainer's first move should be to switch Pokémon, favoring long-term rewards. After the switch, the Water Pokémon has both Water and Fighting attacks (effective against Fire and Normal Pokémon, respectively), and should pick those moves with high priority. This was the strategy used by the hard-coded policy, as advised by human experts.

An example is shown in Figure 2, demonstrating how the converged GIGAθ trainer behaves in the unfair scenario. The trainer prioritizes strategic choices and exploits type advantages, by following the policy described previously. Without any previous domain knowledge, it learned how Fire is vulnerable to Water, Water to Grass, and Normal to Fighting, and makes strategic decisions while using effective moves when possible.

VI. Conclusion

This work described the implementation of a simulator of Pokémon battles, the PBE. Pokémon battling demonstrates interesting properties in terms of competitive AI research. This environment allows for researchers to test various learning objectives. Recent work has shown that interest exists in this type of environment, barely explored in current literature.

With PBE, competitive deep reinforcement learning algorithms were able to learn type-advantage and long term strategies without any *a priori* domain knowledge. Switching to a benched Pokémon which has better typing against the opponent active Pokémon may lead to a more efficient defeat of the opponent Pokémon, even though the short-term reward is negative. WPLθ has been shown to outperform GIGAθ in heavily stochastic environments, but PBE commonly provides agents with a deterministic optimal choice, and GIGAθ was able to learn faster than WPLθ. GIGAθ was also the only algorithm that learned a successful policy with all 18 Pokémon types, achieving a 0.8 normalized score and a 100% win-rate in our test scenario.

Our results are promising, showing that learning Pokémon battles with complete rules may be achievable. Future work will involve adding and testing additional mechanics of Pokémon battles, such as status effects, abilities, and non-damaging moves that influence short and long term rewards. These will then be tested in conjunction with the currently implemented mechanics, to move towards the learning objectives of the Showdown AI Competition.

References

[1] Sherief Abdallah and Victor Lesser. A multiagent reinforcement learning algorithm with non-linear dynamics. *Journal of Artificial Intelligence Research*, 33:521–549, 2008.

[2] Alejandro Alfonso. Pokémon battle. Web: https://pokemon-battle.herokuapp.com/, 2019.

[3] Michael Bowling. Convergence and no-regret in multiagent learning. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 209–216, Cambridge, MA, USA, 2004. MIT Press.

[4] Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'01, pages 1021–1026, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[6] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *Trans. Sys. Man Cyber Part C*, 38(2):156–172, March 2008.

[7] Saurabh Charde. Predicting pokémon battle winner using machine learning. Submitted to conference. Web: shorturl.at/kGRS3, 2019.

[8] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *International Conference for Learning Representations*, 2016.

[9] coyotte508. Pokémon online. Web: http://pokemon-online.eu/, 2019.

[10] Arnaud Durand. Pokémon battle api. Web: https://github.com/DurandA/pokemon-battle-api, 2019.

[11] Vlad Firoiu, William F. Whitney, and Joshua B. Tenenbaum. Beating the world's best at super smash bros. with deep reinforcement learning. *CoRR*, abs/1702.06230, 2017.

[12] Nintendo Game Freak, Creatures Inc. Pokemon series. Web: https://www.pokemon.com, 1996.

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[14] Paul Hallet. Pokéapi. Web: https://pokeapi.co/, 2019.

[15] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

[16] Akshay Kalose, Kris Kaya, and Alvin Kim. Optimal battle strategy in pokemon using reinforcement learning. Web: https://web.stanford.edu/class/aa228/reports/2018/final151.pdf, 2018.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations, San Diego*, 2015.

[18] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM, 1997.

[19] S. Lee and J. Togelius. Showdown ai competition. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 191–198, Aug 2017.

[20] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, volume 157, pages 157–163, 1994.

[21] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

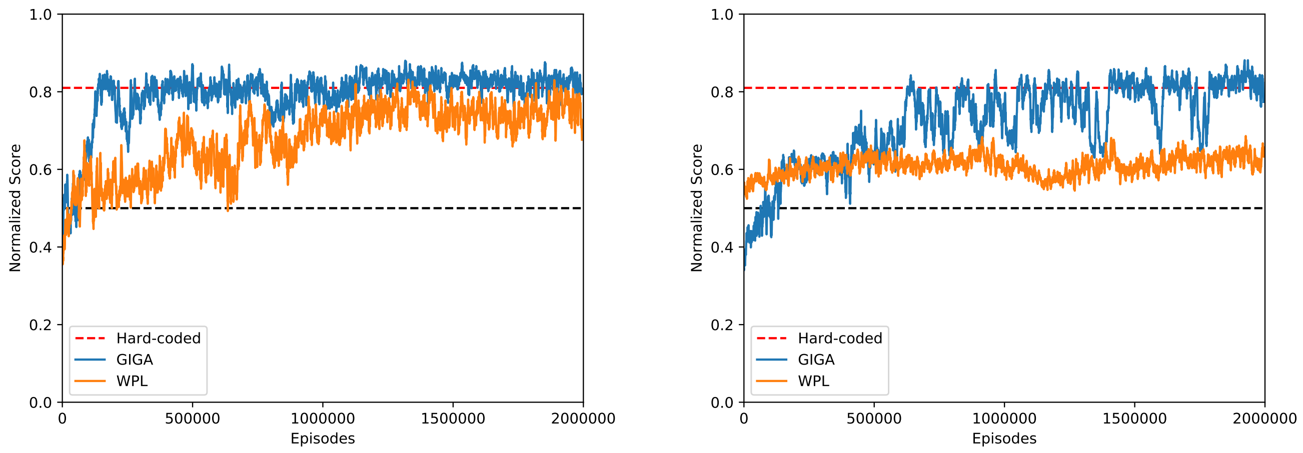

Fig. 1: Results of GIGAθ and WPLθ for the PBE, in an unfair test scenario. The plots represent the normalized score of each algorithm, trained in self-play but tested against a random agent, over training episodes. Because the initial state places the agents in a disadvantageous position, a weak agent (such as a random agent) would obtain a normalized score below 0.5. The plots also show the hard-coded baseline given by expert players.

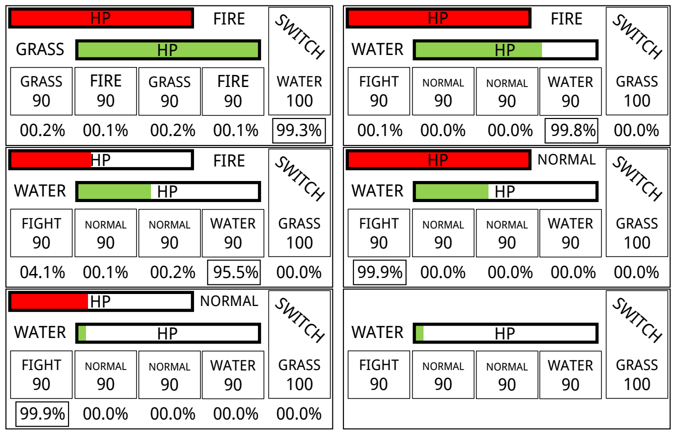

Fig. 2: A converged GIGAθ agent's policy in a fixed test scenario. Initial setting is as follows: Opponent Pokémon is Fire-type, and own type is Grass, and we can observe our Pokémon moves and type of the benched Pokémon. The policy of the agent for each state is shown below, with the chosen action highlighted. The opponent's Pokémon use attacks of their own type every turn, but the initial switch gives the local agent an advantage (switch from Grass-type to Water-type to fight the opponent Fire-type), and it wins the battle with no casualties.

[22] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[24] 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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[25] John F Nash et al. Equilibrium points in n-person games. *Proc. Nat. Acad. Sci. USA*, 36(1):48–49, 1950.

[26] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[27] David Simões, Nuno Lau, and Luís Paulo Reis. Guided deep reinforcement learning in the geofriends2 environment. In *IJCNN 18: Internation Joint-Conference on Neural Networks*, pages 375–381, 2018.

[28] David Simões, Nuno Lau, and Luís Paulo Reis. Mixed-policy asynchronous deep q-learning. In Anibal Ollero, Alberto Sanfeliu, Luis Montano, Nuno Lau, and Carlos Cardeira, editors, *ROBOT 2017: Third Iberian Robotics Conference*, pages 129–140. Springer International Publishing, 2018.

[29] David Simões, Nuno Lau, and Luís Paulo Reis. Multi-agent deep reinforcement learning with emergent communication. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019.

[30] David Simões, Nuno Lau, and Luís Paulo Reis. Multi-agent neural reinforcement-learning system with communication. In Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, and Sandra Costanzo, editors, *New Knowledge in Information Systems and Technologies*, pages 3–12. Springer International Publishing, 2019.

[31] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[32] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[33] Zarel. Pokémon showdown. Web: https://pokemonshowdown.com/, 2019.