Multi-Armed Bandits for Minesweeper: Profiting from Exploration-Exploitation Synergy

Igor Q. Lordeiro and Douglas O. Cardoso

***Abstract*—A popular computer puzzle, the game of Minesweeper requires its human players to have a mix of both luck and strategy to succeed. Analyzing these aspects more formally, in our research we assessed the feasibility of a novel methodology based on Reinforcement Learning as an adequate approach to tackle the problem presented by this game. For this purpose we employed Multi-Armed Bandit algorithms which were carefully adapted in order to enable their use to define autonomous computational players, targeting to make the best use of some game peculiarities. After experimental evaluation, results showed that this approach was indeed successful, especially in smaller game boards, such as the standard beginner level. Despite this fact the main contribution of this work is a detailed examination of Minesweeper from a learning perspective, which led to various original insights which are thoroughly discussed.**

***Index Terms*—Computer Games, Reinforcement Learning, Greedy Policy, UCB, Transfer Learning**

I. Introduction

MINESWEEPER is a single-player computer puzzle game originating from the 1960s. Since its initial release it has evolved in diverse aspects, having many different iterations and being included in various operating systems, most noticeably in members of the Microsoft Windows family. Its small set of rules as well as straightforward gameplay mechanics boosted its popularity up to the point of being considered one of the most successful games ever [1].

This game has been analyzed from a perspective of theory of computation, and its solution was related to NP-completeness [2] and co-NP-completeness [3] according to the premisses considered. Such results support the notion that the game is indeed challenging despite its apparent simplicity. Reinforcement Learning (RL) [4] has been used to tackle general NP-hard problems [5]–[7] as well as to create AI players for classic games such as Battleships, Chess, Shogi and Go [8], [9]. The same goes for Minesweeper, which was approached using RL following some general, frequently used guidelines in this sense [10]. This work is aimed at better exploiting some game properties previously overlooked, which led to a novel RL-oriented modeling whose exploration provided interesting insights and results.

Using Multi-Armed Bandit algorithms [11] and without the aid of any heuristics to pursue this goal, our agents managed to achieve win rates of over 75% and 55% in the beginner and intermediate difficulties, respectively. Though we expected for a purely greedy agent to be unrivaled in the setting enforced by the game of Minesweeper, UCB agents were able to achieve convincing results, only faltering in higher difficulties. While overcoming the UCB agent, the greedy approach also managed to challenge an initial hypothesis of correspondence between knowledge extent and performance in this scenario, as despite its higher win rate, its number of actions learned is significantly lower.

The remainder of this paper is organized as follows. Section II presents the game of Minesweeper, from its basic concepts up to its mechanics and strategic principles. A multifaceted overview of related research can be found in section III. Section IV renders the original ideas we conceived as well as a thorough discussion of them. Practical results obtained from experimenting on the aforementioned ideas are presented in section V, which also explains how performance was assessed. At last, section VI provides some closing remarks and indicates possible continuations of this work.

II. A Brief Game Walk-Through

A Minesweeper game is played using an $m \times n$ board of tiles. There are k hidden mines which are randomly placed just after the always-safe first play/click. Considering the classic Windows 98 version of the game, difficulty levels vary between beginner (8×8 board, 10 mines), intermediate (16×16 board, 40 mines) and expert (16×30 board, 99 mines). Besides these presets the game can be played in a custom format: the notation $r\times c\times b$ is used from now on in this paper to represent a $r \times c$ board with b mines. The player's task is to uncover all tiles that do not feature a mine.

As the player clicks on a tile, a few things can happen: if a mine is uncovered, game over; otherwise, the tile then becomes exposed, showing the number of mines in its eight neighboring tiles; if such number is zero, then the tile becomes blank and all the surrounding tiles become exposed, recursively. It is also possible to place a flag on a tile which is assumed to feature a mine, acting as a marker to prevent clicking on the said tile afterwards. On the other hand, the player can regret placing any flag and remove it.

In a broad sense, rational playing relies on information provided by numbered tiles to deduce where mines are hidden and which tiles are safe. Figure 1 exemplifies a game state and the analysis which could be used to decide the next moves. While some game situations are inherently ambiguous, most of them can be solved based on similar reasoning. Occasionally this strategy needs to be complemented with conjecturing where mines are located and consequently finding contradictions. This clearly suggests to approach the game as

Igor Q. Lordeiro (corresponding author) and Douglas O. Cardoso are with the Department of Computer Engineering, Federal Center for Technological Education Celso Suckow da Fonseca (CEFET-RJ), Petrópolis, RJ, Brazil. E-mails: igor.lordeiro@aluno.cefet-rj.br, douglas.cardoso@cefet-rj.br .

a Constraint Satisfaction Problem (CSP), and indeed diverse works in the literature do likewise in some regard [10], [12].

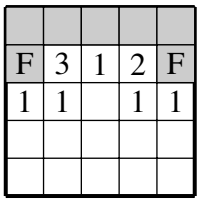


Fig. 1. An ongoing Minesweeper game in a 5×5 board with 4 mines. A pair of mines, each marked by a flag (F), are trivial to detect thanks to the 1-tiles beside them. It is impossible to find the last 2 mines immediately, but it can be inferred that they are located in the 3 leftmost tiles of the first row, according to the 3-tile on position $(2, 2)$. Therefore the last two tiles of the first row are safe, and clicking on them would lead to solving this game.

III. Related Work

Reinforcement learning is a popular technique for creating agents to play games. Its first application in this regard dates back to 1959 with Samuel [13] creating a Checkers agent. More recently, RL has played a defining role on the development of competitive agents for games, with Clementis [8] applying it to Battleships. More notably, Silver, Hubert, Schrittwieser, *et al.* [9] were able to achieve groundbreaking results in Chess, Shogi and Go with AlphaZero. Recent advances also include the work of Jaskowski [14] in the game of 2048, McPartland and Gallagher [15] and Glavin and Madden [16] in First Person Shooter (FPS) games and Pinto and Coutinho [17] in Fighting Games.

Minesweeper automatic solving was first approached by Adamatzky [18] in 1997, who devised a cellular automaton to play it. However the game was studied from a computational complexity perspective only in 2000, when Kaye [2] proved that the problem of assigning mines to covered tiles, which was called Minesweeper consistency problem, is NP-complete. Scott, Stege, and Rooij [3] came to the conclusion that finding whether or not there is at least one covered tile whose mine status could be undoubtedly decided in any game instance, called Minesweeper inference problem, is co-NP-complete.

Following research had its focus mainly on constraint-satisfaction and heuristic methods. Buffet, Lee, Lin, *et al.* [10] used a hybrid of Upper Confidence Trees and Heuristic CSP to achieve win rates of 80.2%, 74.4%, and 38.7% in the beginner, intermediate and expert levels, respectively. Meanwhile Tu, Li, Chen, *et al.* [19] obtained win rates of 81.6%, 78.1%, and 39.6% respectively for the beginner, intermediate and expert levels combining multiple heuristics.

Regarding our approach to Minesweeper, Sutton and Barto [4] and Slivkins [11] didactically introduce the Multi-Armed Bandit (MAB) problem and some of its solutions. Despite being a classic RL problem, some of its real-world applications were assessed recently in the literature: e.g., online advertising, as demonstrated by Pike-Burke, Agrawal, Szepesvári, *et al.* [20] and Chen, Wang, Yuan, *et al.* [21]; resource distribution, studied by Claure, Chen, Modi, *et al.* [22]; and computer networking, shown by Vermorel and Mohri [23].

IV. Our Methodology

This section is organized as follows. Subsection IV-A introduces the proposed approach towards the game of Minesweeper as well as justifies some of our modelling choices. Subsection IV-B displays the differences between the standard MAB problem and Minesweeper, highlighting their implications. Subsection IV-C showcases the considered symmetries and how to obtain them given our modeling of actions. Subsection IV-D presents the game characteristics which we believe to be beneficial to transfer learning.

A. Initial Problem Modeling

A Markov Decision Process (MDP) is a mathematical structure which represents a stochastic process. It can be described as 4-tuple (S, A, p, r) with components defined as follows:

- S is the set of states of the process;
- A is the set of all possible actions, with $A(s) \subseteq A$ being the actions available within state $s \in S$;
- $p : S \times S \times A \to [0, 1]$ is the transition probability function, so that $p(s'|s, a)$ is the probability of reaching state s' from state s as a consequence of action a;
- $r : S \times A \times S \to \mathbb{R}$ is the reward function (also known as signal), so that $r(s, a, s')$ represents the expected reward for taking action a in state s and then reaching state s'.

Straightforwardly, Minesweeper could be modeled as MDP whose states are whole board configurations, defined by mine coordinates and tile conditions (covered, exposed or flagged). In each state unexposed tiles imply possible actions, and there is no uncertainty in the realization of any action: $p : S \times S \times A \to \{0, 1\}$. However, a player is unable to be sure of the state of the game during its course, since mines are hidden. One last aspect to be observed is the fact that there is no benefit in winning the game in as few moves as possible. In other words, this modeling would lead to a deterministic, partially observable, undiscounted MDP.

Now consider, for the sake of argument, that game states could be perfectly acknowledged, so that it would be possible to compute in a tabular fashion $Q(s, a)$, an estimate of how valuable it is to take action a whenever s is the game state. This could in turn be used to always play the best action in any game state. In spite of that, such approach would be computationally intractable even in the beginner level of the game: the number of all possible mine arrangements equals to those of 10-combinations of a set with 64 elements; each tile can be exposed or not, resulting in 2^{64} variations in this sense; therefore there would be over $2^{64} \cdot \binom{64}{10} \sim 10^{30}$ states.

To circumvent this problem, we decided to take inspiration from a human approach of the game: looking at numbered tiles around covered ones and probing for those that allow an assertive decision about hiding a mine or not. Such strategy capitalizes on the fact that in numerous situations the entire board provides as much information as the neighborhood of a tile under consideration for the aforementioned decision. We fulfilled this idea modeling actions as a 10-tuples containing the information of the tiles in a 3×3 portion of the board plus an action target, which indicates an unexposed tile on the

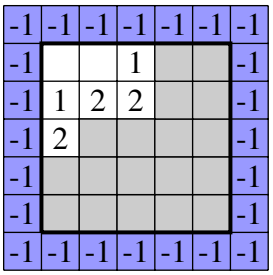


Fig. 2. A framed game board: the blue tiles are out of bounds. In this example a bomb can be trivially found on coordinate (3, 2). This can be inferred based on the 2-tile on (2, 2) and its surroundings, represented by the action (0, 0, 1, 1, 2, 2, 2, C, C, S). 'C' stands for covered tiles. The same is possible based on the 1-tile on (2, 1), as two distinct actions: (-1, 0, 0, -1, 1, 2, -1, 2, C, SE) and (-1, 1, 2, -1, 2, C, -1, C, C, E). The action target (last entry of the tuple) is denoted by cardinal directions abbreviations: 'S' for South, 'SE' for southeast, and so on. The other 5 actions which would also target (3, 2) but do not allow detecting the mine it features are: (-1, 2, C, -1, C, C, -1, C, C, NE), (2, C, C, C, C, C, C, C, C, N), (C, C, C, C, C, C, C, C, C, NW), (2, 2, C, C, C, C, C, C, C, W) and (0, 1, C, 2, 2, C, C, C, C, SW).

border of the frame that is to be played. Figure 2 illustrates this concept.

An upper bound for the number of actions in this setting is $8 \cdot 12^8 \sim 10^{10}$, considering that there are 8 possible action targets and each of the other 8 tiles can be exposed, exhibiting a value from 0 to 8, or be covered and with no flag, or be flagged, or even be outside the game board, totaling 12 cases. However, it is important to point that not all of these variations can indeed happen in a Minesweeper game. Beyond that, because some of them are symmetric, what is thoroughly discussed in subsection IV-C, the number of truly distinct actions is considerably below this limit.

Another interesting fact regarding this action modeling, relying on neighborhood information only, is that it induces discovering useful game patterns: actions which are surely safe or always lead to revealing a mine, regardless of the rest of the board in which one of this actions can be realized. In other words this shifts the focus from $Q(s, a)$ towards $Q(a)$, the action value, taking no account of the board configuration and therefore greatly reducing the computational cost of playing the game. This action-only perspective combined with the deterministic and undiscounted attributes of Minesweeper are the cornerstones of its approach as a MAB problem.

B. Distinctions From Standard MAB

In a standard MAB setting the goal is to maximize the total reward received from repeatedly choosing from a fixed set of options one action to perform and receiving a corresponding reward next, which is randomly defined according to some hidden stationary probability distribution. Despite some resemblance, this differs in a few ways from the problem presented by Minesweeper. Thus we aimed at adapting existing MAB solutions in order to make the best use of such peculiarities of the game while preserving the guarantees they provide.

At first sight Minesweeper deviates from the target of reward maximization, as the game does not have a scoring system. However, finding which actions have the best chance of being safe and which are likely to reveal mines can be related to identifying those that consistently provide good rewards. These rewards can be defined according to the only possible action outcomes: uncovering of a safe tile, or a mine explosion. And since every action which does not immediately terminate the game is positive, accomplishing as many of these as possible surely leads to the ultimate objective. With all this in mind we established the reward function $r(s, a, s') = 1$ or -1 if action a reveals a mine or not, respectively.

This last reasoning brings attention to the exploration-exploitation dilemma in this context: while exploration allows refining the expectation of an action being safe or not, it risks ending the game prematurely due to its sheer nature; on the other hand exploitation tends to prolong games, creating opportunities to learn about actions whose occurrence is more frequent in mid- and late-game situations and actually winning. Moreover, as the set of actions at each time step can be distinct, what contrasts with MAB defaults, exploration inevitably happens, even when explicitly avoided.

These characteristics enabled the use of an always-greedy policy to successfully play the game, combined with the following action value setup: initially, for every action a, $Q(a) = -1$ and $N(a) = 0$, where $N(a)$ is the number of times the action was performed; whenever a happens, $N(a)$ is incremented by 1, and then $Q(a) + \frac{1}{N(a)}(R - Q(a))$ is assigned to $Q(a)$, where R is the reward resulting from the action. In this setup every play which was never tried is considered to be perfectly safe and, therefore, as valuable as possible as an action. This is specially significant to the greedy approach as it enforces some exploration and, with the use of the greater action count $N(a)$ as a tie-breaker when choosing the next action, also creates a suitable balance of exploration and exploitation for the agents.

The last peculiarity tackled by our modeling presents itself in the form of flags. While it is possible to play without using this feature, simply by not clicking known mine tiles, flags offer the opportunity of conveying the information that a tile is probably a mine to other actions, whereas simply not clicking the tile does not. This is illustrated in figure 3.

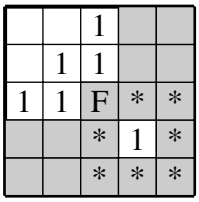


Fig. 3. In this hypothetical board state, the tiles marked by an asterisk can be deemed safe due to the flag placed in (3, 3). From the point of view of our modeling, the action which ultimately resulted in that flag being placed enabled to deduce that the *-tiles are safe.

Deciding when to place a flag instead of unveiling a tile is the first question to be answered in order to enable its use. Fortunately the already established reward and action value functions provide the necessary support to such decision: instead of always choosing the available action of lowest value to uncover a tile, the action with greatest value can be preferred for flag placement if it has the greatest absolute value. This is coherent with the fact that a low action value indicates

consistency about the fact that an action is safe just as a high action value with respect to the expectation that an action would reveal a mine. Since winning is impossible if a safe tile is flagged, the following directive was used: once the number of flags surpasses the number of mines, the flagged tile with the lowest action value is forcibly unveiled.

Still regarding flags, while uncovering a tile will immediately portray whether this was a positive action or not, placing a flag does not provide any instant feedback, what hinders fixing a mistaken negative perception of an action. For that reason, in our approach learning also occurs just after the game is finished, regardless of the result being a win or a loss: during the game, all actions which ultimately resulted in a flag being placed are recorded; when the game ends this record is revisited, checking whether indeed there was a mine where the flags were placed or not, receiving the adequate reward. Such delayed learning is only possible thanks to the fact that the mines are revealed when the game is over. To take full advantage of such information, all actions which were passed over in the turn when the game was finished also have their values updated as if they were chosen among all others, based on the information that they were hiding a mine or not.

C. Symmetries

From the perspective of our modeling, we consider 2 distinct cases for symmetries, determined by the tile to be played through an action. The first, called the diagonal case, occurs when the tile which is the action target lies in one of the 4 corners of the 3×3 portion of the board the action covers. The second, called the cardinal case, takes place when the action target is perpendicularly above, below, or besides the central tile of the 3×3 action configuration. The intrinsic differences between these two cases which substantiate the need to treat them separately are illustrated in figure 4.

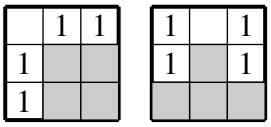


Fig. 4. This example shows why it is impossible to transform a diagonal case into a cardinal case or vice-versa. The non-center tiles of the action configuration on the left are shifted clockwise to transform it into the opposite case. This creates an impossible action configuration, as a 0-tile (blank) cannot share a side with a covered tile.

For the diagonal case, 8 symmetric action configurations are possible, as illustrated by figure 5. As for the cardinal case, another 8 possible symmetric action configurations are also considered, as presented in figure 6. In both cases the first 4 configurations can be obtained by rotating the depicted grids around the center tile by $90°$. The last 4 configurations are the result of mirroring the first 4 along the axis containing the center tile and the action target.

Taking into account the considered symmetries, the ceiling for the number of action configurations has an eightfold decrease, becoming $12^8 \sim 10^9$. While this is indeed a substantial difference, this ceiling still significantly exaggerates how many

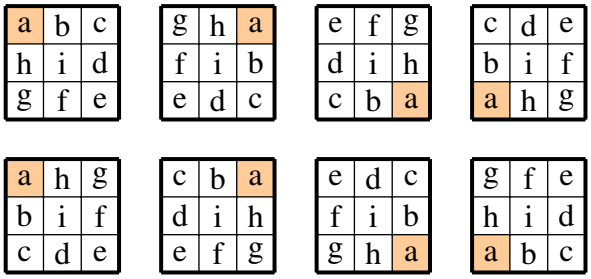


Fig. 5. Symmetries in the diagonal case. The first row exemplifies the action rotations, while the second one shows their respective inverses. The targeted tile in each action indexes has orange color.

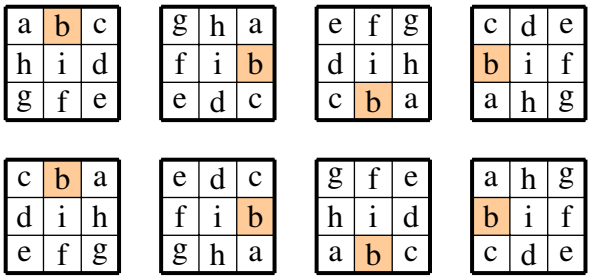


Fig. 6. Symmetries in the cardinal case. The first row exemplifies the action rotations, while the second one shows their respective inverses. The targeted tile in each action indexes has orange color.

action configurations can happen in a Minesweeper game. For example, it is not possible for an 8-tile to neighbor anything but covered tiles or for a 7-tile to neighbor 3 or more numbered tiles. These and many other configurations which are considered in the 12^8 count would never be found in any real game.

D. Transfer Learning and Training Optimization

Having at least 8 mines is a necessary condition for a game to be able to feature every valid action configuration. However, this is not a sufficient condition: for example, a 1-tile cannot happen in a degenerate $3 \times 3 \times 8$ game. Despite this fact there is no doubt that sufficiently bigger settings allow the occurrence of any action configuration. This is just the case for standard Minesweeper levels, which have different board dimensions and number of mines, but the possible actions to be performed by an agent in the beginner level would be the same in the intermediate or expert levels. On the other hand, how probable is the presence of an action configuration during a game in each of these settings is not the same. This inspires the use of transfer learning: to consider cross-settings training for best overall performance.

Analyzing the influence of game parameters on learning is the way to look for the ideal training scenario. Although the number of mines is one of game attributes which have the most direct influence on learning and playing, how big the board is provides some perspective about this amount. Consequently mine density is the prime factor when accounting for action configuration frequencies, as higher densities tend to produce action configurations with higher numbered tiles. This makes

most agents trained in low-density settings to perform poorly in high-density games, and vice-versa. Moreover, mine density is an indicator of how difficult a game setting is, and its variation affects not only win rate but also the number of plays per game and, ultimately, the number of actions learned in a predefined number of games.

This last assertion points to another aspect of training optimization, that is learning efficiency. Straightforwardly, smaller boards generally equate in shorter games while larger boards tend to the contrary. And an extremely high or low mine density leads to quick games, as each click has a higher chance of clearing the whole board or exposing a mine. At last, a $16 \times 30 \times 99$ game is computationally more expensive to set up and play than a $8 \times 8 \times 10$ one. Balancing these points is the key to obtain solid knowledge about a substantial variety of the most frequent action configurations, playing as few games as possible, with board dimensions as small as possible. Considering that training is constrained not by the number of games but time, although with a few hindrances, agents trained in settings closer to standard beginner level yielded better results overall than intermediate and expert rivals.

V. Experimental Evaluation

Each individual experiment comprised a series of episodes, which were complete Minesweeper games. In subsection V-A all experiments used a $8 \times 8 \times 10$ setting, the standard beginner level. Specific board sizes, and mine amounts or densities are indicated in other subsections. For the majority of experiments each agent was trained in 10^6 episodes and some meaningful statistics were then reported. The only exception of this rule were the experiments of subsection V-C, in which the agents were trained during 10^6 episodes but then tested in varied scenarios for 10^4 episodes each time. The main effectiveness metric used is the win rate, what is coherent with the fact that the game has no scoring system. In this regard is worthy noting that the win rate of an agent steadily increases during its training, tending to what could be seen as a *true* win rate, which can be estimated in tests realized after training win rate is considered stable. The source code of the experiments is available upon request by e-mail.

A. Initial Impressions

The first experiments targeted to tune MAB algorithms in order to achieve win rates as high as possible, so that the top performing contestants would be further analyzed in other experiments. A summary of the results in this preliminary selection can be seen in figure 7. The highest win rate was that of the strictly greedy agent, followed by an ϵ-greedy one with $\epsilon = 0.01$ and then by tied UCB agents with $c = 0.01$ and 0.1. Although these greedy and ϵ-greedy agents had the highest win rates, the performance of these UCB agents was enough to indicate that exploration is not a problem in itself, and that it could possibly pay off in the long run considering how the curves of the last pair differ from those of the first.

Aside from win rate, another statistic which one could assume to be important in this setting is the number of actions recorded. Figure 8 illustrates this statistic regarding the greedy

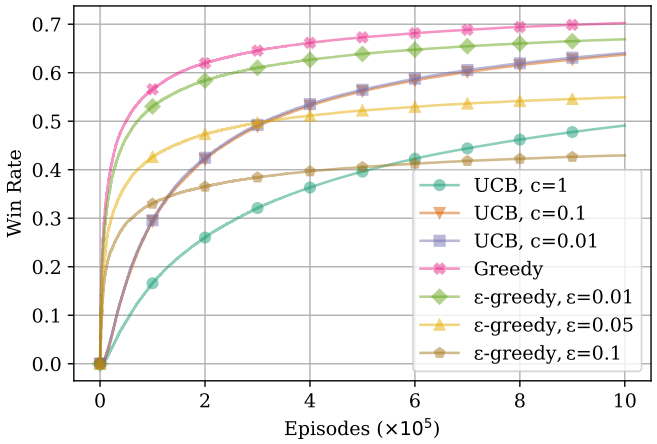


Fig. 7. Win rate evolution during training of a variety of agents. The shape of the curves establishes a clear distinction between UCB and (ϵ-)greedy agents.

and UCB ($c = 0.1$) agents. While it would be expected for the greedy agent to learn about fewer actions, as its strategy relies exclusively in exploitation, it was first hypothesized that by lasting longer in games it would compensate such trait, coming to possess information about a larger variety of actions. Eventually this would lead to getting the number of actions recorded close to that of the UCB agent. But it is undeniable that this was not the case: despite the lower win rate, the UCB agent recorded almost twice the number of actions the greedy agent registered; regarding *perfect* actions (i.e., those with $Q(a) \in \{-1, 1\}$, whose outcome could be fairly expected to be deterministic) the case is similar. In short, learning about more actions does not guarantee to convert such vaster knowledge into higher win rates.

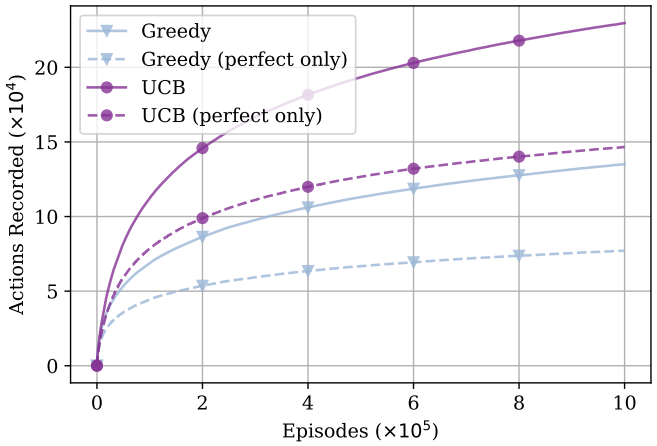
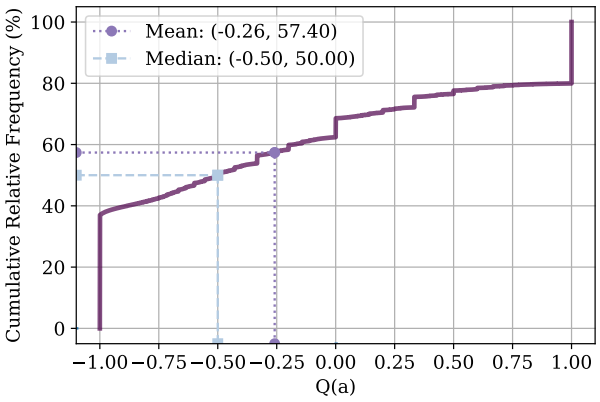


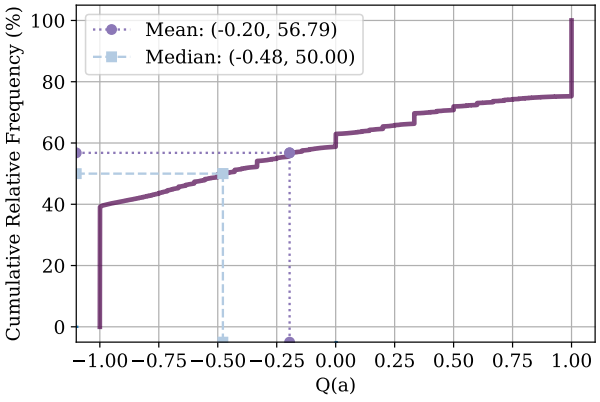
Fig. 8. Comparison of actions learned over episodes between the two best performing agents from the previous experiment. The "perfect" tag refers to actions with an action-value of either -1 or 1, so that there is no uncertainty about the result of the concretization of any of them.

A deeper look at the knowledge accumulated during training could provide an explanation of this phenomenon. Figure 9 presents the plot of the empirical cumulative distribution function (ECDF) of the action values of the selected agents.

Notably, most actions recorded by both agents are considered perfect, but it should be taken into account that most of these actions have this status only because of having a single past execution, what guarantees such condition. The majority of the presumed perfect actions are safe, which is reasonable as it is likely for the number of safe plays needed to win a game to be greater than the number of mines in it. On the other hand the non-perfect actions represent the innate game of chance that Minesweeper can be. Overall both agents have a quite similar profile with respect to $Q(a)$.



(a) Greedy



(b) UCB

Fig. 9. ECDF plots of the action values of the selected agents.

While the distributions of the action values of the selected agents are almost indistinguishable, the action counts provide a more interesting perspective. As shown in table I, on average the actions registered by the greedy agent have a greater count than those related to UCB. Since the action counts represent how many times the actions were performed, it could be seen as an indication of experience and confidence about the action values. Thus it is reasonable to affirm that the greedy agent generally would have a more precise notion of the expected outcome each of the actions it recorded. And based on its superior results compared to those of the UCB agent it could be claimed that the last was farther from finding the balance between knowledge length and depth than its rival.

TABLE I
AVERAGES OF THE ACTION COUNTS OF THE SELECTED AGENTS

Perfect Action?	Agent	
	Greedy	UCB
Yes	99.2432	51.2799
No	582.6260	435.4277
Both	306.7781	190.1957

To assess the impact flags and symmetries have on the selected agents, a collection of experiments was realized in which handicapped versions of the agents were used: one just avoided risky tiles instead of explicitly flagging them; another considered symmetric actions different, learning about each of them separately; the third and last one combined the characteristics of the first two. The results are portrayed in figure 10. While disabling symmetries only sets back the progress of the agents, disabling flags seems to greatly hinder their learning, with both agents' win rate reaching less than 12%. This result emphasizes the importance of placing flags for the proposed approach, showing that even in the beginner level neglecting a detail as such can substantially harm success in the task at hand. When symmetries are disabled, as actions which could be seen as equivalent are then considered distinct, there is a substantial increase in the number of actions registered, perfect or not.

B. Tinkering With Game Mechanics

Having scouted the way an agent interacts with a fixed Minesweeper setting, the following experiments were focused on observing the effects game attributes such as board dimensions and mine density have on learning, aiming at finding out how to use them to improve training. For this purpose 2 collections of experiments were carried out: one varying board width and height from 3 to 10 while approximately maintaining a constant mine density of 0.15625, which is the same of the standard beginner level; and the other with a fixed board width of 10 columns while varying its width from 3 to 10 as well as mine density from 0.14 to 0.2. These experiments were performed using purely greedy agents, not only because it was the top performer in the initial evaluation but also because its simpler mechanics facilitates realizing the desired analysis.

Figure 11 confirms that board dimensions indeed influence win rate even with a constant mine density. This can be explained straightforwardly: a larger board would feature a greater number of mines spread across its cells and consequently would probably require a greater number of plays to be defeated; as the number of mines increase there is a higher likelihood of risky, non-perfect actions, and eventually they could become the only available options. Moreover, it can be noticed that the standard beginner and intermediate levels have the same mine density, but the board of the latter is 4 times bigger than that of the former, so that winning a game of the intermediate level can be compared to winning 4 games of the beginner level in a row.

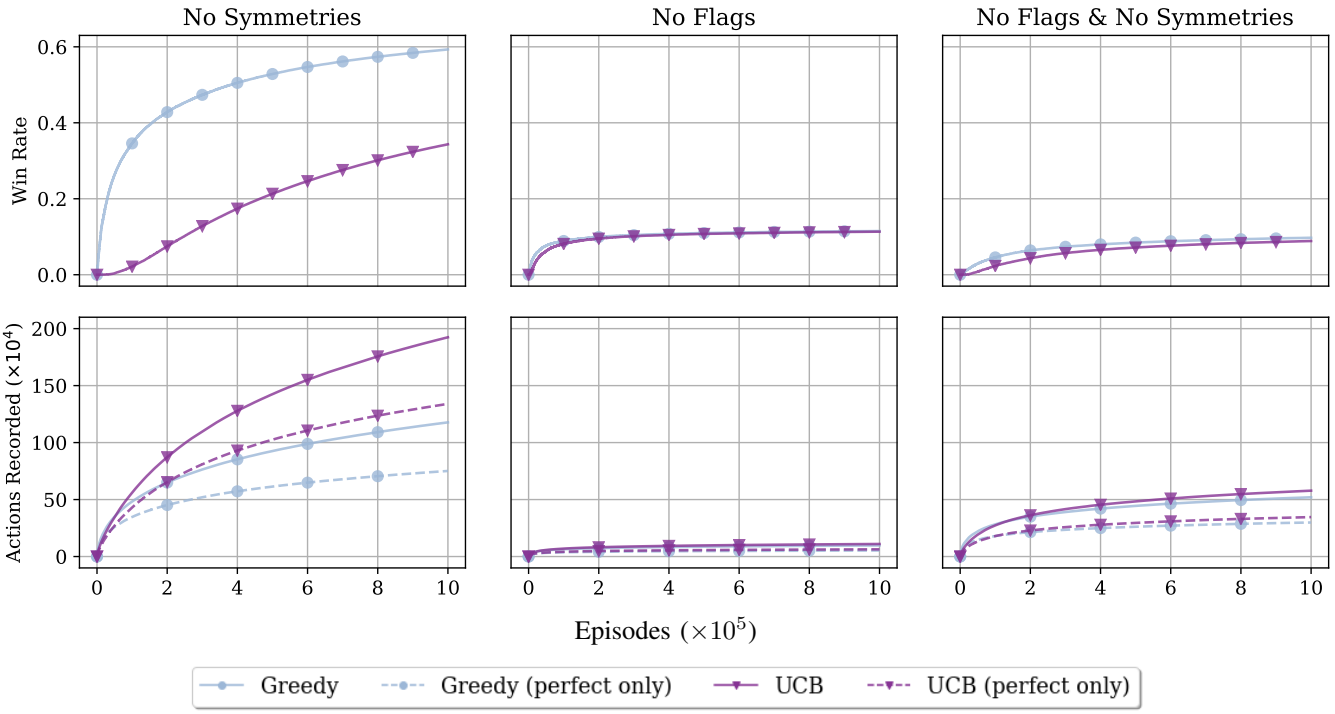


Fig. 10. Performance of the selected greedy and UCB agents when trained without using flags, considering symmetries, or both.

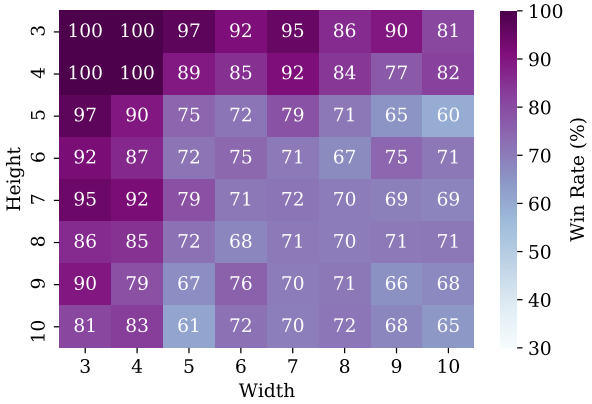


Fig. 11. Heatmap displaying the win rate of greedy agents trained on varied board dimensions with the number of mines defined rounding the value obtained considering a fixed mine density of 0.15625.

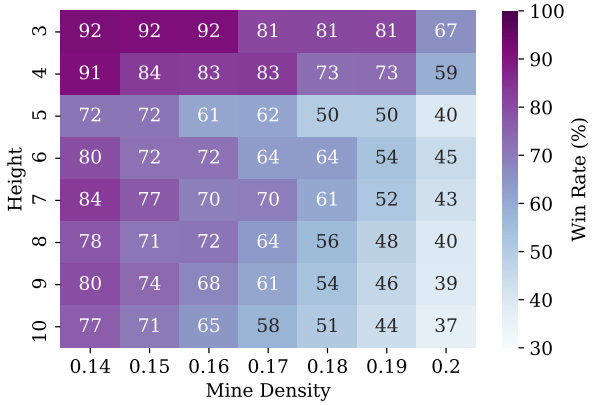


Fig. 12. Heatmap displaying the win rate of greedy agents trained on varied board heights and mine densities with a fixed width of 10 columns. The number of mines is defined rounding the value obtained considering the just mentioned attributes.

Figure 12 also presents fairly interpretable results, as by increasing board height the win rate decreases but by increasing mine density the same happens more significantly: on average the decrease produced by the first was of 26.1% versus 35.3% of the second. Though expected, these results further accentuate the overwhelming importance of mine density when discussing win rate, since an increase by 0.06 in this attribute could result in a drop in win rate of at least 25%, considering the results just reported. The combined increase of mine density along with the board dimensions is what makes the expert level brutally harder, as shown and discussed next.

C. Transfer Learning Results

Of the game attributes which were covered in the just reported experiments, mine density was the most intriguing, considering how it could be used for training optimization: to modify game difficulty while keeping board dimensions unaltered sparked expectations for agents which could be trained in higher density settings to thrive in lower density ones. The caveats of such use of mine density are that the win rate is highly sensitive to changes in it, and that if the disparity between the mine density of the board an agent is trained on and the board he is tested on is too large the agent could not be able to successfully employ its knowledge in the test setting.

Table II displays the discrepancy between agents trained with the same board dimensions but with different mine densities. By simply adding 3 mines from the standard of 10 mines in the beginner level the training win rate plummets from 70.2% to 41.9% and reached as low as 11% with 17 mines. However, noticing that such reduction of training effectiveness does not necessarily produce the same effect on post-training tests, as shown in the bottom 3 rows of the same table, inspired a deeper exploration of such fact.

Still in the same context, it was contemplated the idea of training using not one but a combination of mine densities and along with it arose the question of whether or not it presented any benefits over the more conventional training using a single setting, with the ultimate goal of producing an agent better fit for the intermediate and expert levels. Then an agent was made to train in a 8×8 board with 10 to 13 mines, $25 \cdot 10^4$ episodes for each number of mines, totaling 10^6 episodes. These mine amounts were chosen targeting to cover as close as possible the mine densities of the standard levels from beginner up to expert. Though this idea could look promising from start, results displayed in table II show that an agent subjected to this alternative training routine does not present any true advantages when compared to another one trained in a higher mine density from the beginning.

Training on higher mine densities can indeed create agents better suited for cross-settings evaluation, but it does not entail that the higher the density the better the agent performs from a transfer learning perspective. Table II also shows that despite the increase in training mine density, there is a decrease in win rate in all standard levels when trained in an setting exaggeratedly dense such as $8 \times 8 \times 17$. These results indicate that mimicking mine density is not enough when creating a successful agent based on transfer learning. And as suggested by the results of the $8 \times 8 \times 15$ greedy agent, it can be valid to train in a density higher than the one targeted in order to create a margin for what we suspect to be a form of compensation for the additional difficulty inherent in larger boards.

Table III presents the results of UCB agents in experiments in the same fashion of those realized with greedy agents. More than for the simple sake of completeness, reporting these results is interesting as they show how UCB agents exhibit the exact same behavior of their greedy rivals, profiting from training in a higher density setting for a better overall performance. This was enough to make the performance of the $8 \times 8 \times 15$ UCB agent in the beginner level the second best of all tested options. Nevertheless, the disparity between the greedy and UCB agents only became more apparent with mine density increase, favoring the first. In the end the superiority of the $8 \times 8 \times 15$ greedy agent above all its rivals can be safely stated: every sample of game outcomes (0 for lose, 1 for win) of all experiments can be seen as derived from a normally distributed population according to D'Agostino-Pearson tests with a significance level of 0.01; and there is no overlap between respective Student t-test 99% confidence intervals.

VI. Conclusion

Minesweeper provides a challenging task for any agent to master. Past works with this goal mostly relied on CSP and heuristic approaches, which could be considered incontrovertible choices taking into account some game aspects. Ultimately they could provide very good win rates, the most basic success benchmark in this context. However, this subject appeared to be far from being exhausted. We suspected that if the problem modeling contemplated some other key game properties, then RL could be fruitful in this setting, not only in the sense of winning but also for effectively investigating the game from a novel perspective. This motivated the development of the proposed MDP modeling as well as the adaptations of classic MAB algorithms ϵ-greedy and UCB.

The best performing agent resulting from our methodology was able to achieve a win rate of over 70% while learning Minesweeper from scratch in 10^6 games of the standard beginner level. Providing such a detailed description of the conditions to which the agent was subjected, instead of just the win rate, is uttermost important since we are more interested in learning than in winning. The same goes for the fact that the just mentioned agent is purely greedy, and that it outdid its rivals despite learning in significantly more succinct fashion. Moreover, we sought to optimize training inspired in transfer learning, what resulted in the just mentioned best agent obtaining win rates of 76.96%, 57.94% and 4.13% in the beginner, intermediate and expert levels, respectively.

The main obstacle we found while experimenting with Minesweeper was the pure brutality of large boards' difficulty when tackled by our approach. Training over 10^6 games in a $8 \times 8 \times 15$ setting took 14 hours, while playing 10^4 $16 \times 30 \times 99$ games took almost 7.5 hours. This makes it virtually impossible to train on larger boards, which led us to use transfer learning as an alternative. But still, while results on the intermediate level are satisfactory, the same cannot be said about the expert level. In future works we hope to perfect the balance of learning and game settings to define an agent capable of handling larger boards to an acceptable degree.

Acknowledgments

Igor Q. Lordeiro (Scientific Initiation Scolarship/Bolsista PIBIC) would like to thank CEFET-RJ for financial support.

References

[1] R. Cobbett, *The most successful game ever: A history of Minesweeper*, 2009. [Online]. Available: https://www.techradar.com/news/gaming/the-most-successful-game-ever-a-history-of-minesweeper-596504 (cit. on p. 1).

[2] R. Kaye, "Minesweeper is NP-complete," *The Mathematical Intelligencer*, vol. 22, no. 2, pp. 9–15, 2000. [Online]. Available: https://doi.org/10.1007/BF03025367 (cit. on pp. 1, 2).

[3] A. Scott, U. Stege, and I. van Rooij, "Minesweeper may not be NP-complete but is hard nonetheless," *The Mathematical Intelligencer*, vol. 33, no. 4, pp. 5–17, 2011. [Online]. Available: https://doi.org/10.1007/s00283-011-9256-x (cit. on pp. 1, 2).

[4] R. Sutton and A. Barto, *Reinforcement learning: An introduction*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2018. [Online]. Available: https://books.google.com.br/books?id=sWV0DwAAQBAJ (cit. on pp. 1, 2).

TABLE II
WIN RATE 99% CONFIDENCE INTERVALS OF SELECTED GREEDY AGENTS TRAINED AND TESTED IN VARIED SCENARIOS

Scenario	Game Setting				
	$8 \times 8 \times 10$	$8 \times 8 \times (10..13)$	$8 \times 8 \times 13$	$8 \times 8 \times 15$	$8 \times 8 \times 17$
Training	$\mathbf{0.7025 \pm 0.0011}$	0.5515 ± 0.0012	0.4193 ± 0.0012	0.2504 ± 0.0011	0.1105 ± 0.0008
Beginner	0.7350 ± 0.0113	0.7520 ± 0.0111	0.7550 ± 0.0110	$\mathbf{0.7696 \pm 0.0108}$	0.7570 ± 0.0107
Intermediate	0.4409 ± 0.0127	0.5206 ± 0.0128	0.5707 ± 0.0127	$\mathbf{0.5794 \pm 0.0127}$	0.5341 ± 0.0128
Expert	0.0028 ± 0.0013	0.0168 ± 0.0033	0.0269 ± 0.0041	$\mathbf{0.0413 \pm 0.0051}$	0.0341 ± 0.0046

TABLE III
WIN RATE 99% CONFIDENCE INTERVALS OF SELECTED UCB AGENTS TRAINED AND TESTED IN VARIED SCENARIOS

Scenario	Game Setting	
	$8 \times 8 \times 10$	$8 \times 8 \times 15$
Training	$\mathbf{0.6372 \pm 0.0012}$	0.1839 ± 0.0009
Beginner	0.7289 ± 0.0114	$\mathbf{0.7644 \pm 0.0109}$
Intermediate	0.2699 ± 0.0114	$\mathbf{0.4453 \pm 0.0128}$
Expert	0.0000 ± 0.0000	$\mathbf{0.0062 \pm 0.0020}$

[5] F. Yang, T. Jin, T. Liu, X. Sun, and J. Zhang, "Boosting dynamic programming with neural networks for solving NP-hard problems," in *Proceedings of The 10th Asian Conference on Machine Learning, ACML 2018, Beijing, China, November 14-16, 2018*, J. Zhu and I. Takeuchi, Eds., ser. Proceedings of Machine Learning Research, vol. 95, PMLR, 2018, pp. 726–739. [Online]. Available: http://proceedings.mlr.press/v95/yang18a.html (cit. on p. 1).

[6] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=Bk9mxlSFx (cit. on p. 1).

[7] K. Abe, Z. Xu, I. Sato, and M. Sugiyama, "Solving NP-hard problems on graphs by reinforcement learning without domain knowledge," *CoRR*, vol. abs/1905.11623, 2019. [Online]. Available: http://arxiv.org/abs/1905.11623 (cit. on p. 1).

[8] L. Clementis, "Supervised and reinforcement learning in neural network based approach to the battleship game strategy," in *Nostradamus 2013: Prediction, Modeling and Analysis of Complex Systems, Nostradamus conference 2013, Ostrava, Czech Republic, June 2013*, I. Zelinka, G. Chen, O. E. Rössler, V. Snásel, and A. Abraham, Eds., ser. Advances in Intelligent Systems and Computing, vol. 210, Springer, 2013, pp. 191–200. [Online]. Available: https://doi.org/10.1007/978-3-319-00542-3_20 (cit. on pp. 1, 2).

[9] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018. [Online]. Available: https://science.sciencemag.org/content/362/6419/1140 (cit. on pp. 1, 2).

[10] O. Buffet, C.-S. Lee, W.-T. Lin, and O. Teytuad, "Optimistic heuristics for minesweeper," in *Advances in Intelligent Systems and Applications - Volume 1*, R.-S. Chang, L. C. Jain, and S.-L. Peng, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 199–207 (cit. on pp. 1, 2).

[11] A. Slivkins, "Introduction to multi-armed bandits," *Found. Trends Mach. Learn.*, vol. 12, no. 1-2, pp. 1–286, 2019. [Online]. Available: https://doi.org/10.1561/2200000068 (cit. on pp. 1, 2).

[12] Y. Tang, T. Jiang, and Y. Hu, "A minesweeper solver using logic inference, CSP and sampling," *CoRR*, vol. abs/1810.03151, 2018. [Online]. Available: http://arxiv.org/abs/1810.03151 (cit. on p. 2).

[13] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM J. Res. Dev.*, vol. 3, no. 3, pp. 210–229, 1959. [Online]. Available: https://doi.org/10.1147/rd.33.0210 (cit. on p. 2).

[14] W. Jaskowski, "Mastering 2048 with delayed temporal coherence learning, multistage weight promotion, redundant encoding, and carousel shaping," *IEEE Trans. Games*, vol. 10, no. 1, pp. 3–14, 2018. [Online]. Available: https://doi.org/10.1109/TCIAIG.2017.2651887 (cit. on p. 2).

[15] M. McPartland and M. Gallagher, "Reinforcement learning in first person shooter games," *IEEE Trans. Comput. Intellig. and AI in Games*, vol. 3, no. 1, pp. 43–56, 2011. [Online]. Available: https://doi.org/10.1109/TCIAIG.2010.2100395 (cit. on p. 2).

[16] F. G. Glavin and M. G. Madden, "Adaptive shooting for bots in first person shooter games using reinforcement learning," *IEEE Trans. Comput. Intellig. and AI in Games*, vol. 7, no. 2, pp. 180–192, 2015. [Online]. Available: https://doi.org/10.1109/TCIAIG.2014.2363042 (cit. on p. 2).

[17] I. P. Pinto and L. R. Coutinho, "Hierarchical reinforcement learning with monte carlo tree search in computer fighting game," *IEEE Trans. Games*, vol. 11, no. 3, pp. 290–295, 2019. [Online]. Available: https://doi.org/10.1109/TG.2018.2846028 (cit. on p. 2).

[18] A. Adamatzky, "How cellular automaton plays minesweeper," *Applied Mathematics and Computation*, vol. 85, no. 2, pp. 127–137, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0096300396001178 (cit. on p. 2).

[19] J. Tu, T. Li, S. Chen, C. Zu, and Z. Gu, "Exploring efficient strategies for minesweeper," in *The Workshops of the The Thirty-First AAAI Conference on Artificial Intelligence, Saturday, February 4-9, 2017, San Francisco, California, USA*, ser. AAAI Workshops, vol. WS-17, AAAI Press, 2017. [Online]. Available: http://aaai.org/ocs/index.php/WS/AAAIW17/paper/view/15091 (cit. on p. 2).

[20] C. Pike-Burke, S. Agrawal, C. Szepesvári, and S. Grünewälder, "Bandits with delayed, aggregated anonymous feedback," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, J. G. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 4102–4110. [Online]. Available: http://proceedings.mlr.press/v80/pike-burke18a.html (cit. on p. 2).

[21] W. Chen, Y. Wang, Y. Yuan, and Q. Wang, "Combinatorial multi-armed bandit and its extension to probabilistically triggered arms," *J. Mach. Learn. Res.*, vol. 17, 50:1–50:33, 2016. [Online]. Available: http://jmlr.org/papers/v17/14-298.html (cit. on p. 2).

[22] H. Claure, Y. Chen, J. Modi, M. F. Jung, and S. Nikolaidis, "Multi-armed bandits with fairness constraints for distributing resources to human teammates," in *HRI '20: ACM/IEEE International Conference on Human-Robot Interaction, Cambridge, United Kingdom, March 23-26, 2020*, T. Belpaeme, J. Young, H. Gunes, and L. D. Riek, Eds., ACM, 2020, pp. 299–308. [Online]. Available: https://doi.org/10.1145/3319502.3374806 (cit. on p. 2).

[23] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Machine Learning: ECML 2005, 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005, Proceedings*, J. Gama, R. Camacho, P. Brazdil, A. Jorge, and L. Torgo, Eds., ser. Lecture Notes in Computer Science, vol. 3720, Springer, 2005, pp. 437–448. [Online]. Available: https://doi.org/10.1007/11564096_42 (cit. on p. 2).