Acta Polytechnica **54**(5):333–340, 2014
doi:10.14311/AP.2014.54.0333

available online at http://ojs.cvut.cz/ojs/index.php/ap

ALTERNATIVE SELECTION FUNCTIONS FOR INFORMATION SET MONTE CARLO TREE SEARCH

Viliam Lisý

Agent Technology Center, Department of Computer Science, FEE, Czech Technical University in Prague
correspondence: viliam.lisy@agents.fel.cvut.cz

Abstract. We evaluate the performance of various selection methods for the Monte Carlo Tree Search algorithm in two-player zero-sum extensive-form games with imperfect information. We compare the standard Upper Confident Bounds applied to Trees (UCT) along with the less common Exponential Weights for Exploration and Exploitation (Exp3) and novel Regret matching (RM) selection in two distinct imperfect information games: Imperfect Information Goofspiel and Phantom Tic-Tac-Toe. We show that UCT after initial fast convergence towards a Nash equilibrium computes increasingly worse strategies after some point in time. This is not the case with Exp3 and RM, which also show superior performance in head-to-head matches.

Keywords: Monte Carlo Tree Search, imperfect information game, selection function, Regret matching.

1. Introduction

Monte Carlo Tree Search (MCTS) is a family of sample-based tree search algorithms that has recently led to a significant improvement in the quality of state-of-the-art solvers for perfect information problems, such as the game of Go [1] or domain independent planning [2]. The main idea of Monte Carlo Tree Search is to run a large number of randomized simulations of the problem and learn the best actions to choose from the experience. The earlier simulations are generally used to create statistics that help to guide the later simulations to more important parts of the search space and decide on the best action to take in the current state of the game. The core component of the algorithm determining which action to choose next and what statistics to collect is called a *selection function*.

Inspired by the success of MCTS in perfect information games, the algorithm has recently also been adapted for imperfect information games [3–5]. Games with imperfect information are fundamentally more complicated than perfect information games, for several reasons. The most important complication is that the optimal strategies in imperfect information games may require the players to make randomized decisions. For example, in the well-known game of Rock-Paper-Scissors, none of the available actions can be considered optimal. Playing any of the actions all the time can always be exploited by the opponent, and the optimal strategy against a rational opponent is to play each action with the same probability. Another important complication is the strong inter-dependency between the strategies in different parts of the game. A player often does not know the exact state of the game during the match, and the probability of the game being in the individual possible states depends on the opponent's strategy in previous decisions in the game, which can depend on the optimal strategy in any other decision in the game.

Game theory provides means to deal with all these complications, but previous attempts to adapt MCTS to imperfect information games generally did not evaluate the properties of the strategies from the perspective of game theory. The algorithms were developed mainly on the basis of heuristics and analogies with the perfect information case.

In this paper, we analyze various selection functions in Information Set Monte Carlo Tree Search (IS-MCTS) [3]. We show that the most commonly used selection function – UCT ([6]) – does not allow the algorithm to converge to good strategies, and even causes the strategies to get worse with more computation time. We further evaluate two additional selection functions. Exp3 [7], which has previously been used in MCTS mainly to handle simultaneous moves [3, 8, 9], and Regret Matching [10], previously evaluated in the context of simultaneous move games [9], but never used for MCTS in generic imperfect information games. In an imperfect information variant of the game of Goofspiel and Phantom Tic-Tac-Toe, we show that these alternative selection strategies allow IS-MCTS to converge closer to the Nash equilibrium strategy and perform better in mutual matches.

The following section introduces the basic game-theoretic concepts necessary to describe the IS-MCTS algorithm along with the existing and novel selection functions in Section 3. Afterwards, we continue with an experimental evaluation in Section 4, and then we conclude the paper.

2. Definitions and Background

We focus on domains that can be modelled as two-player zero-sum extensive-form imperfect-information games.

2.1. Extensive form games

We adapt the definition of the Extensive-form game (EFG) from [11].

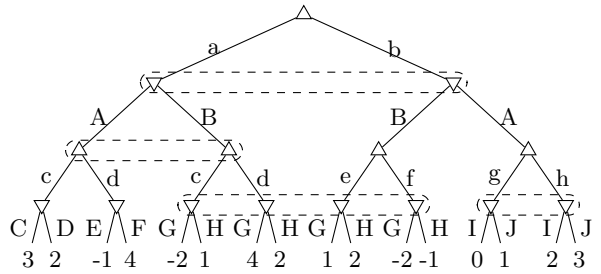

Figure 1. Example of a zero-sum imperfect-information extensive-form game.

Definition 1 (Extensive-form game)**.** An extensive form game with imperfect information consists of:

- $\mathcal{N}$ is the set of players including the *nature* player (c) that represents the dynamics of the game;
- for each $i \in \mathcal{N}$, $\mathcal{A}_i$ is the set of actions available to player i;
- $\mathcal{H}$ is the set of possible states of the game, each corresponding to a *history* of actions performed by all players;
- $\mathcal{Z} \subseteq \mathcal{H}$ is the set of terminal game states or histories;
- $\mathcal{P} : \mathcal{H} \setminus \mathcal{Z} \to \mathcal{N}$ is a function assigning to each non-terminal state a player that selects an action in the state;
- $\mathcal{A} : \mathcal{H} \setminus \mathcal{Z} \to 2^{\mathcal{A}_i}$ is a function assigning to each non-terminal game state the actions applicable by the acting player;
- $\mathcal{T} : \mathcal{H} \times A_i \to \mathcal{H} \cup \mathcal{Z}$ is the transition function realizing one move of the game;
- $u_i : \mathcal{Z} \to \mathbb{R}$ for each $i \in \mathcal{N} \setminus \{c\}$ are the utility functions of the players defined only on the terminal states of the game;
- $\mathcal{I}_i$ for player i represents the player's imperfect information about the game. It is a partition of $\mathcal{H}_i = \{h \in \mathcal{H} : \mathcal{P}(h) = i\}$ termed *information sets*. Each information set $I \in \mathcal{I}_i$ represents the set of histories that are indistinguishable for player i. Therefore, we naturally extend $\mathcal{A}(I) = \mathcal{A}(h)$ for some $h \in I$.

The game starts with the empty history $\emptyset$. In each history h, player $P(h)$ selects an action $a \in \mathcal{A}(h)$. After the action is performed, the game proceeds to history $h' = \mathcal{T}(h, a)$, often also denoted ha. The game ends when it reaches a terminal history $h \in \mathcal{Z}$. Each player $i \in \mathcal{N}$ receives the payoff $u_i(h)$. A *zero-sum* game is a two player game ($|\mathcal{N} \setminus \{c\}| = 2$), such that for each terminal history $h \in \mathcal{Z}$, the reward for one player is a loss for the other player ($u_i(h) = -u_{-i}(h)$).

Extensive form games can be represented by a game tree, such as the one in Figure 1. The set of histories $\mathcal{H}$ are the nodes in the tree and $\mathcal{Z}$ are the leaves. We denote the nodes where the first player selects an action ($\mathcal{H}_1$) by $\triangle$, and the nodes of the second player (minimizing the utility of the first player) by $\triangledown$. In this example, the action sets of the players are $\mathcal{A}_1 = \{a, b, \ldots, h\}$, $\mathcal{A}_2 = \{A, B, \ldots, J\}$. We denote the information sets as the ellipses around the tree nodes. After the history $h = aA$ player $\triangle$ decides about the next move from $\mathcal{A}(h) = \{c, d\}$, but in the game, he has exactly the same information as if $h' = aB$ was the current state of the game. The numbers in the leaves denote the utility of the first player.

A *pure strategy* in an extensive form game is a function that assigns to each information set $I \in \mathcal{I}_i$ of player i an action from $\mathcal{A}(I)$. Each pair of pure strategies then naturally defines the utility function as the expected value of playing this pair of strategies. The expectation is taken over the actions in the chance nodes, in which the action is selected based on a commonly known probability distribution. The set of *mixed strategies* of an extensive form game is defined as the set of all probability distributions over the pure strategies. In games where players do not forget the actions they performed, a theorem proposed by Khun [12] allows us to use the following equivalent, but much more compact, representation of the mixed strategies. A *behavioral strategy* assigns to each information set a probability distribution over the available actions. If it is not specified otherwise, we mean by a strategy in an extensive form game the behavioral strategy.

We denote the set of all mixed strategies of player $i \in \mathcal{N}$ by Σ_i. We term a vector of one strategy for each player $\sigma \in \Pi_{i \in \mathcal{N}} \Sigma_i$ a *strategy profile* and we denote σ_i the strategy of player i and we denote σ_{-i} the strategy of the opponent of i. $\Delta(\mathcal{A}_i)$ is the set of all probability distributions over $\mathcal{A}_i$. u_i can be naturally extended to mixed strategies as the expectation over the pure strategies.

Definition 2 (Best response)**.** For strategy profile σ, we define a *best response* of player i to the opponent's strategy σ_{-i} as the strategy

$$\mathsf{br}(\sigma_{-i}) \in \arg\max_{\sigma'_i \in \Delta(\mathcal{A}_i)} u_i(\sigma'_i, \sigma_{-i}).$$

One of the most fundamental solution concepts in game theory is the Nash equilibrium [11].

Definition 3 (Nash equilibrium)**.** A strategy profile $\sigma^* \in \Sigma$ is a *Nash Equilibrium* if

$$u_i(\sigma_i, \sigma^*_{-i}) \leq u_i(\sigma^*) \quad \forall i \in \mathcal{N}, \quad \forall \sigma_i \in \Sigma_i.$$

In words, in a Nash equilibrium, each player plays the best response to the strategies of the other players. In zero-sum games, a Nash equilibrium is a very appealing strategy to play. It prescribes a (generally randomized) strategy that is optimal in several aspects. It is a strategy that gains the highest expected reward against its worst-case opponent, even if the opponent knows the strategy played by the player in advance. Moreover, in the zero-sum setting, even if the opponent does not play rationally, the strategy still guarantees at least the reward it would have

gained against the rational opponent. This guaranteed reward is termed the *value* of the game.

The distance of a strategy from a Nash equilibrium can be measured in terms of *exploitability* (e.g., [13]).

Definition 4 (Exploitability)**.** The exploitability of strategy σ_i is

$$\text{expl}(\sigma_i) = v_i - u_i(\sigma_i, \mathsf{br}(\sigma_i))$$

where v_i is the value of the game for player i. The exploitability of strategy profile σ is

$$\text{expl}(\sigma) = u_i(\sigma_i, \mathsf{br}(\sigma_i)) - u_i(\mathsf{br}(\sigma_{-i}), \sigma_i).$$

In this paper, we compute the exploitability using the best response function described in [14].

3. Information Set Monte Carlo Tree Search

Information Set Monte Carlo Tree Search (IS-MCTS) is a Monte Carlo tree search variant for imperfect information games. Fundamentally very similar algorithms have previously been formulated in two different ways. The formulation that is easier to understand is presented in [4]. The MCTS iterations are performed on the complete game tree of the extensive form games (i.e., all possible states on the game). However, the statistics for the selection algorithm, such as UCT, are collected for the whole information set. If any of the nodes in the information set is reached, the selection algorithm is used to select the next move and, subsequently, the statistics stored by this algorithm are updated by the result of the simulation. In the tree expansion step, the authors in [4] suggest adding a single node to the extensive form tree of the game.

A very similar algorithm is presented in [15], in [3] as "Multiple-Observer Information Set Monte Carlo tree search" and in [5] as "Multiple Monte Carlo Tree Search". The pseudo-code for a single iteration of the algorithm is presented in Algorithm 1. It starts in the root node of the game tree and descends the game tree towards a terminal state. The tree nodes are stored in the memory and common statistics are stored for all nodes in each information set. If the function is called with a terminal state, it just returns its utility for the first player (line 1). If nature selects an action in the current node (line 2), it is selected from the commonly known nature distribution and the algorithm is called recursively on the resulting state (line 4). Otherwise, the node is added to memory if it is not already present (line 5). The statistics for the information set it belongs to are accessed (line 6) and used to make the decision about the action to select (line 8). This action is then executed on the game state, producing the following game state, which is used in a recursive call of the function (line 9). This process is continued until a state belonging to a new information set is reached (line 10). This is the end of the selection stage, and the expansion consists of creating new

IS-MCTS(h)

```
1: if h is terminal (h ∈ Z) then return u_1(h)
2: if h is a chance node (P(h) = c) then
3:     a = random chance action from A_c(h)
4:     return IS-MCTS(T(h, a))
5: if h not in memory then add h to memory
6: IS = information set for state h
7: if IS is not null then
8:     a = select(IS)
9:     v = IS-MCTS(T(h, a))
10: else
11:    add new IS for h to memory
12:    a = select(IS)
13:    v = simulate(T(h, a))
14: update(IS, a, v)
15: return v
```

Algorithm 1. Information Set Monte Carlo Tree Search

(empty) statistics for the information set (line 11). The specific data structures depend on the selection function. At this point, an action is selected by the selection function and the simulation is executed to estimate the quality of the following position (lines 12–13). The information sets accessed during the iteration are updated by the result of the simulation (line 14) when returning from the recursion, and the next iteration can start. Iterations are repeated until a given time budget is spent.

The functions *select* and *update* in the pseudo-code can be implemented by a suitable selection function, such as UCT (with the negative of the received value for the opponent), and *simulate* can be either completely random, or a domain dependent simulation, as in perfect information MCTS. In the following section, we discuss a possible selection function to be used with the algorithm.

During an actual match, the player using this algorithm does not always start iterations from the root state of the game, but it rather maintains the current information set in the form of a collection of all states that can be the current state of the game. After the player using IS-MCTS selects an action, it applies the action to all of these states to determine the possible following states. When the opponent executes some action in the game, it executes all of the opponent's applicable actions in the current information set and keeps only the resulting states that generate the same observations for the player.

In the search from an inner information set during a match, the player chooses for each iteration the initial state uniformly at random from all states in the current information set [3].

3.1. Selection Functions

3.1.1. Upper Confidence Bounds

The selection function that was suggested for IS-MCTS in both [4] and [3] is the same modification of

Input: K – number of actions; C – exploration parameter
1: $\forall_i n_i = 0, \bar{x}_i = 0$
2: **for** $n = 1, 2, \ldots$ **do**
3: $\quad i = \arg\max_i \bar{x}_i + C\sqrt{\frac{2 \ln n}{n_i}}$
4: $\quad$ Use action i and receive reward r
5: $\quad \bar{x}_i = \frac{n_i \bar{x}_i + r}{n_i + 1}$
6: $\quad n_i = n_i + 1$

Algorithm 2. UCT: Upper Confidence Bounds algorithm for selection in Monte Carlo Tree Search.

Input: K – number of actions; γ – exploration parameter
1: $\forall_i \hat{x}_i = 0$
2: **for** $t = 1, 2, \ldots$ **do**
3: $\quad \forall_i p_i = \frac{exp(\frac{\gamma}{K}\hat{x}_i)}{\sum_{j=1}^{K} exp(\frac{\gamma}{K}\hat{x}_j)}$
4: $\quad p'_i = (1-\gamma)p_i + \frac{\gamma}{K}$
5: $\quad$ Use action I_t from distribution p' and receive reward r
6: $\quad \hat{x}_{I_t} = \hat{x}_{I_t} + \frac{r}{p'_{I_t}}$

Algorithm 3. Exp3: Exponential weights for Exploration and Exploitation algorithm for selection in MCTS.

UCB1 [16] that was successful in perfect information MCTS [6]. We present the algorithm in Algorithm 2 and further refer to it as UCT. The algorithm maintains the mean of the rewards received for each action $\bar{x}_i$ and the number of times each action has been used n_i. It first uses each of the actions once (the term with zero in the denominator is defined as ∞) and then decides what action to use based on the size of the one-sided confidence interval on the reward computed based on the Cheroff-Hoeffding bounds (line 3). We follow the suggestion from [17] and break ties on line 3 randomly.

The strategy output as the solution for the information set after all simulations are the action use times n_i normalized to sum to one.

3.2. Exponential Weights for Exploration and Exploitation

UCT is a successful selection function for perfect information problems, but it has been shown to converge to an exploitable strategy in a simultaneous move game [18], which is a special case of imperfect information games. Therefore [3, 8] and [5] propose the use of an alternative selection function Exp3, which can be modified to guarantee convergence to the optimal solution in single-stage simultaneous move games.

Exp3 stores the estimates of the cumulative reward of each action over all iterations, even in the case that the action was not selected. In the pseudo-code in Algorithm 3, we denote this value for action i by $\hat{x}_i$. It is initially set to 0 on line 1. In each iteration, a probability distribution p is created proportionally to the exponential of these estimates. The distribution is combined with a uniform distribution with probability γ to ensure sufficient exploration of all actions (line 4). After an action is selected and the reward is received from the recursive call of IS-MCTS, the estimate for the performed action is updated using importance sampling (line 6). The reward is weighted by one over the probability of using the action, in order to reach the correct value in expectation.

Input: K – number of actions; γ_t – non-increasing sequence of real numbers
1: $\forall i\ R_i = 0$
2: **for** $t = 1, 2, \ldots$ **do**
3: $\quad \forall i\ R_i^+ = \max\{0, R_i\}$
4: $\quad$ **if** $\sum_{j=1}^{K} R_j^+ = 0$ **then**
5: $\quad\quad \forall i\ p_i = 1/K$
6: $\quad$ **else**
7: $\quad\quad \forall i\ p_i = \frac{R_i^+}{\sum_{j=1}^{K} R_j^+}$
8: $\quad p'_i = (1-\gamma)p_i + \frac{\gamma}{K}$
9: $\quad$ Use action I_t from distribution p' and receive reward r
10: $\quad \forall i\ R_i = R_i - r$
11: $\quad R_{I_t} = R_{I_t} + \frac{r}{p'_{I_t}}$

Algorithm 4. RM: Regret matching variant for selection in MCTS.

The strategy output as the solution for the information set after all simulations are performed is the mean of strategies p over all iterations. In the implementation of the algorithm, we use the numerically more stable form of the equation in line 3 proposed in [3].

3.3. Regret Matching

Regret matching is a general procedure originally developed for playing known general-sum matrix games in [10]. The algorithm computes, for each action in each step, the regret for not playing another fixed action every time the action has been played in the past. The action to be played in the next round is selected randomly with probability proportional to the positive portion of the regret for not playing the action. The regret matching procedure in [10] requires exact information about the utility values in the game matrix as well as the action selected by the opponent in each step. In [19], the authors relax these requirements. Instead of computing the exact values for the regrets, the regrets are estimated in a similar way as the cumulative rewards in Exp3. As a result, the modified regret matching procedure can be used as the selection function in IS-MCTS.

We present the algorithm in Algorithm 4. The algorithm stores the estimates of the regrets for not taking action i in all time steps in the past in variables R_i. In lines 3-7, it computes the strategy for the current time step proportionally to the positive part of the regrets. The uniform strategy is added similarly

to the case of Exp3 in line 8. This ensures exploration and causes the addition in line 11 to be bounded. Lines 10 and 11 update the cumulative regrets using importance sampling.

The main computational advantage of this procedure over Exp3 is that it requires only simple division instead of computing the expensive exponential function for each action in each iteration. When used in an MCTS algorithm, this allows substantially more iterations to be performed in the same time budget. The strategy output as the solution for the information set after all simulations are performed is the mean of strategies p over all iterations.

4. Experimental Evaluation

This section first presents two imperfect information games that we use in an evaluation. Afterwards, we analyze the dependence of the speed of convergence of IS-MCTS and the eventual distance from a Nash equilibrium on the used selection function. Finally, we evaluate how these convergence properties translate to the quality of actual game playing. In the whole section, we use hand-tuned values of the exploration parameters $C = 2$ for UCT, $\gamma = 0.1$ for Exp3 and RM. All experiments were performed in a unified publicly available codebase [20].

4.1. Experimental Domains

Goofspiel. Goofspiel is a simultaneous move card game often used to evaluate AI algorithms. The game is played with three identical decks of cards. Each deck contains cards of values $\{0, \ldots, (n-1)\}$ and belongs to one of the players, including nature. The deck for nature is shuffled at the beginning of the game. In each round, nature reveals the top card from its deck. Each player selects any of their remaining cards and places it face down on the table so that the opponent does not see the card. Afterwards, the cards are turned face up and the player with the higher card wins the card revealed by nature. The card is discarded in case of a draw. At the end, the player with the higher sum of nature cards wins the whole game. In the results, we use utilities $1/0/-1$ for win/draw/loss and count a draw as half win half lose in the win rates. We use the imperfect information variant of this game introduced by Lanctot [21] for evaluation. It introduces two modifications. First, in each round, the players only learn who won or lost the round, but not the bid played by the opponent. Second, both players know that the cards in nature's deck are sorted in decreasing order.

Phantom Tic-Tac-Toe. Phantom Tic-Tac-Toe is a blind variant of the well-known game of Tic-Tac-Toe. The game is played on a 3×3 board, where two players (cross and circle) attempt to place 3 identical marks in a horizontal, vertical, or diagonal row to win the game. The player who achieves this goal first wins the game. In the blind variant, the players are unable to observe their opponent's moves and each player only knows that the opponent made a move and it is her turn. If a player tries to place her mark on a square that is already occupied by an opponent's mark, the player learns this information and can place the mark in some other square. The uncertainty in phantom Tic-Tac-Toe makes the game large ($\approx 10^{10}$ nodes [22]). In addition, since one player can try several squares before a move is successful, the players do not necessarily alternate in making their moves. This rule makes the structure of the information sets rather complex, and since the player never learns how many attempts the opponent actually performed, a single information set can contain nodes at a different depth in the game tree.

4.2. Convergence to Nash Equilibrium

First, we focus on the speed of convergence in a small variant of Goofspiel with 6 cards $(0, 1, \ldots, 5)$. We measure the ability of the algorithms to approximate a Nash equilibrium strategies of the complete game by the sum of exploitabilities of both players' strategies (see Section 2).

We compare the IS-MCTS algorithm with three different selection functions: UCT, Exp3, and RM. The results are the means of 20 runs of each algorithm. Due to the different selection and update functions, the algorithms differ in the number of iterations per second. RM is the fastest, with more than 5.9×10^4, while UCT with computing the square roots and random tie breaking has around 3.9×10^4 iterations per second, and Exp3 computing the exponential functions has around 3.4×10^4 iterations. Figure 2 presents the exploitability of the algorithms run from the root state for 30 minutes. Note that the x-scale is logarithmic. The exploitability of UCT starts decreasing fairly quickly, but after approximately 20 seconds of computation it starts increasing again. The lowest error achieved by UCT is 0.39, reached after 25 seconds of computation. The variants of the IS-MCTS algorithm with Exp3 and RM converge slower at the beginning, but eventually achieve smaller error than UCT. After 500 seconds of computation, the error of Exp3 is 0.41 and the error of RM is 0.27.

The game of Phantom Tic-Tac-Toe has approximately 10^{10} terminal states, which makes it difficult to compute the exploitability of the strategies in the whole game. Therefore, we initially focus on a simpler version of the game with the first move enforced to be to the square in the center of the board. The first player has only this action available, and the second player can also first play only this action, which reveals the position of the first player's mark and allows the second player to move again. The second move of the second player and all the following moves can then be any legal moves in Phantom Tic-Tac-Toe.

The number of iterations per second in this restricted game is similar to the previous game. RM makes the most iterations (9.5×10^4), because the

0.1 s	UCT	EXP3	RM	RAND
UCT	62.9 (2.6)	55.6 (2.1)	62.7 (2.5)	84.0 (2.1)
EXP3	74.5 (2.2)	62.8 (1.6)	74.8 (2.0)	88.0 (1.9)
RM	70.6 (1.7)	60.0 (1.6)	73.1 (2.1)	87.8 (1.4)
RAND	15.7 (2.2)	8.2 (1.6)	10.3 (1.8)	49.7 (3.0)
1 s	UCT	EXP3	RM	RAND
UCT	61.5 (2.7)	54.0 (2.1)	64.5 (2.5)	84.2 (2.1)
EXP3	74.5 (2.2)	61.8 (1.5)	75.7 (2.0)	87.8 (1.9)
RM	72.8 (2.3)	59.2 (1.7)	75.2 (2.1)	88.8 (1.9)
RAND	14.9 (2.1)	8.7 (1.7)	10.6 (1.8)	48.7 (3.0)

Table 1. Win rates of the row player against different algorithms in Imperfect Information Goofspiel with 6 cards with 0.1 (top) and 1 (bottom) second of computation per move. The number in brackets indicates the 95 % confidence interval.

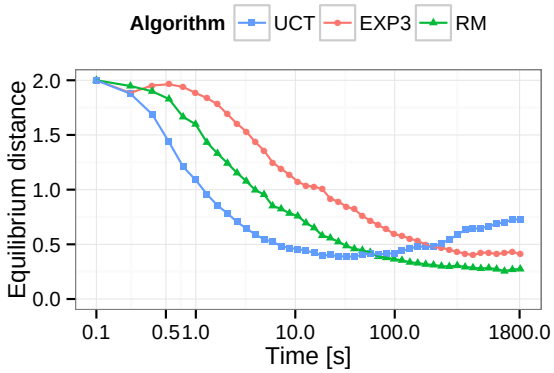

Figure 2. Convergence in the game root in Imperfect Information Goofspiel with 6 cards.

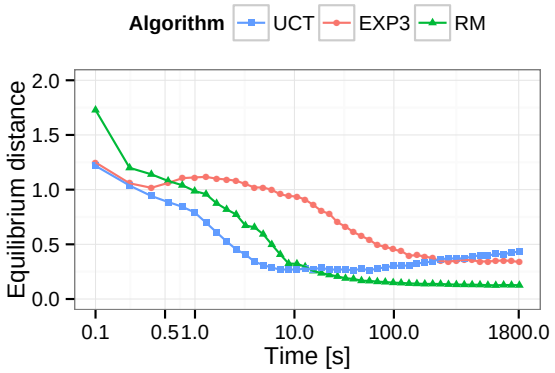

Figure 3. Convergence and game playing comparison of different algorithms on Phantom Tic-Tac-Toe, in which the first move of each player is forced to be to the middle square.

game initially has a higher number of applicable actions and RM uses the simplest functions to compute the probability of selecting each action. UCT performs significantly fewer iterations (5.1×10^4), which is probably caused by the actions often having very similar values. UCT needs to iterate through the actions multiple times to select an action. Exp3 is again the slowest, with 3.4×10^4 iterations per second.

The convergence of the algorithms that are run from the root of the game is presented in Figure 3. UCT first quickly reduces the exploitability of the strategy and then gradually makes the strategy more exploitable after 10 seconds of computation. The minimum exploitability achieved by UCT is 0.27. Shortly after 10 seconds of computation, initially the worst RM becomes the best algorithm and, after the full 30 minutes, it converges to exploitability of 0.13. Exp3 is clearly the worst algorithm between the first and the hundredth second.

4.3. Head to Head Matches

After analyzing the convergence of the strategies computed by the sampling algorithms, we evaluate how this property translates to the actual performance of the algorithms in head-to-head matches. All presented results are an average of 1000 matches. We first evaluate the small game used for computation of the exploitability from the previous section to see the connection to the exploitabilities, and then we focus on substantially larger games to evaluate practical applicability.

Table 1 presents the win rates of the algorithms in mutual matches in Imperfect Information Goofspiel with 6 cards per deck. The table on the top presents the results with 0.1 second per move and the table on the bottom presents the results with 1 second per move, but they are very similar. The first important observation is that even though the game is symmetric and the first player does not have any advantage, all IS-MCTS variants perform much better as the first player (rows). The reason is the asymmetry of the game model in the form of EFG. Even though in reality the players choose an action simultaneously, the game models this fact as a sequential decision with hidden information. As a result, the second player has substantially larger information sets than the first player. If the search is run from a larger information set, it is generally more important to have a good approximation of the probability of the individual states being the actual state of the game. The second player is at a large disadvantage here.

IS-MCTS with Exp3 wins the most from the first

1 s	UCT	EXP3	RM	RAND
UCT	70.0 (2.8)	67.7 (2.9)	61.0 (3.0)	90.6 (1.8)
EXP3	79.5 (2.5)	63.2 (2.8)	66.4 (2.9)	95.6 (1.2)
RM	73.8 (2.7)	68.2 (2.8)	67.5 (2.9)	92.8 (1.5)
RAND	6.2 (1.5)	4.0 (1.2)	4.9 (1.3)	49.0 (3.0)

TABLE 2. Win rate of different algorithms on Imperfect Information Goofspiel with 13 cards. The number in brackets indicates the 95% confidence interval.

0.1 s	UCT	EXP3	RM	RAND
UCT	85.9 (1.7)	84.8 (1.7)	82.3 (1.8)	95.6 (1.1)
EXP3	87.2 (1.6)	88.1 (1.5)	85.3 (1.6)	96.6 (1.0)
RM	87.7 (1.6)	88.1 (1.4)	85.2 (1.5)	96.0 (1.0)
RAND	50.5 (3.0)	46.1 (3.0)	48.0 (3.0)	71.4 (2.7)
1 s	UCT	EXP3	RM	RAND
UCT	86.5 (1.8)	86.3 (1.7)	84.0 (1.8)	95.2 (1.1)
EXP3	88.5 (1.6)	88.5 (1.5)	87.0 (1.6)	96.2 (1.0)
RM	90.0 (1.4)	87.4 (1.5)	85.2 (1.5)	96.3 (1.0)
RAND	52.0 (3.0)	48.9 (3.0)	46.2 (2.9)	70.9 (2.7)

TABLE 3. Win rate of different algorithms on full Phantom Tic-Tac-Toe. The number in brackets indicates the 95% confidence interval.

position and loses the least from the second position. This is a surprising result, as Exp3 has the slowest convergence. Apparently, it quickly reaches a strategy that is exploitable, but performs well against the other opponents in the tournament.

The game playing performance of the algorithms in the large game of Imperfect Information Goofspiel with 13 cards in each deck and one second of computation per move is presented in Table 2. In the large game, the performance of the IS-MCTS variants is similar to the performance on the smaller version. Exp3 wins significantly the most against UCT, but RM loses less against UCT and itself from the second position. This makes it hard to select the better selection function between Exp3 and RM, but both of them are clearly better than UCT in this game.

Since all the algorithms seem to perform very well even in the complete game of Phantom Tic-Tac-Toe, we do not present the evaluation on the smaller game with the enforced first move here. We assume that the differences between the algorithms would be even smaller there. The results in Table 3 show an even larger imbalance between the results achieved from the first and the second position in the game. This time, it is not caused only by the disadvantage of larger information sets, but also by the fact that the game is not balanced on its own. When both players play the optimal strategy, the player who moves first wins 83% of the games. We computed this values by the double-oracle algorithm [14] implemented in our framework.

With 0.1 second per move (top), the performance of RM and Exp3 is practically identical from the first position, but RM loses less on the second position, which is consistent with its generally lower exploitability over the experiments. UCT generally performs the worst. With 1 second per move, the performance of all algorithms is more similar to each other. RM still wins most often and loses least often against UCT, but the mutual matches with Exp3 are more balanced. Either of the algorithms wins 87% of matches from the first position against each other. However, when the algorithms play against themselves, RM loses less from the second position, which makes it the more suitable algorithm for this setting.

5. Conclusions

We have studied the influence of selection functions on the performance of Monte Carlo Tree Search in imperfect information games. We have evaluated the most standard UCT selection along with the less common Exp3 and novel Regret Matching selection in a unified framework on two different games. To the best of our knowledge, this is the first direct comparison of the effect of selection functions on the performance of MCTS in imperfect information games.

We show that none of the proposed selection functions allows the algorithm to converge very close to the Nash equilibrium, and even after 30 minutes of computation (approximately 10^8 iterations), the distance from an equilibrium was still larger than 0.1. Consistently in both evaluation domains, the quality of the strategy produced with UCT first decreased and then started to increase again, showing that IS-MCTS with UCT does not have the desirable anytime property that more computation time yields better results. This is also the case with Exp3 at the very

beginning, but the novel RM selection monotonously converges towards an equilibrium in our experiments.

The superior performance of RM selection was also confirmed in head-to-head matches among the algorithms. In both domains used for evaluation, UCT performed the worst from the evaluated selection functions. RM was always among the best, but in some cases was slightly outperformed by Exp3. The good performance of Exp3 in the matches was not supported by low exploitability in the convergence experiments, which indicates that even though the algorithm performed well against the tested opponents, there are other opponents that are likely to perform much better against this algorithm, but not against RM.

Acknowledgements

This work was supported by the Czech Science Foundation (grant P202/12/2054). Access to the computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the "Projects of Large Infrastructure for Research, Development, and Innovations" programme (LM2010005) is highly appreciated.

References

[1] S. Gelly, D. Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence* **175**(11):1856–1875, 2011. doi:10.1016/j.artint.2011.03.007.

[2] T. Keller, P. Eyerich. Prost: Probabilistic planning based on uct. In *ICAPS*. 2012.

[3] P. I. Cowling, E. J. Powley, D. Whitehouse. Information set monte carlo tree search. *Computational Intelligence and AI in Games, IEEE Transactions on* **4**(2):120–143, 2012. doi:10.1109/TCIAIG.2012.2200894.

[4] M. Ponsen, S. de Jong, M. Lanctot. Computing approximate Nash equilibria and robust best-responses using sampling. *Journal of Artificial Intelligence Research* **42**:575–605, 2011. doi:10.1613/jair.3402.

[5] D. Auger. Multiple tree for partially observable monte-carlo tree search. In *Applications of Evolutionary Computation*, pp. 53–62. Springer, 2011.

[6] L. Kocsis, C. Szepesvári. Bandit based monte-carlo planning. In J. Frnkranz, T. Scheffer, M. Spiliopoulou (eds.), *Machine Learning: ECML 2006*, vol. 4212 of *Lecture Notes in Computer Science*, pp. 282–293. Springer Berlin Heidelberg, 2006. doi:10.1007/11871842_29.

[7] P. Auer, N. Cesa-Bianchi, Y. Freund, R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J Comput* **32**(1):48–77, 2003. doi:10.1137/S0097539701398375.

[8] O. Teytaud, S. Flory. Upper confidence trees with short term partial information. In *Applications of Evolutionary Computation*, vol. 6624 of *Lecture Notes in Computer Science*, pp. 153–162. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-20525-5_16.

[9] M. Lanctot, V. Lisý, M. H. M. Winands. Monte Carlo tree search in simultaneous move games with applications to Goofspiel. In *Computer Games Workshop at IJCAI 2013*, vol. 408 of *Communications in Computer and Information Science (CCIS)*, pp. 28–43. Springer, 2014.

[10] S. Hart, A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica* **68**(5):1127–1150, 2000. doi:10.1111/1468-0262.00153.

[11] M. Osborne, A. Rubinstein. *A course in game theory*. MIT press, 1994.

[12] H. W. Kuhn. Extensive games and the problem of information. *Contributions to the Theory of Games* **2**(28):193–216, 1953.

[13] M. Johanson, K. Waugh, M. Bowling, M. Zinkevich. Accelerating best response calculation in large extensive games. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pp. 258–265. AAAI Press, 2011. doi:10.5591/978-1-57735-516-8/IJCAI11-054.

[14] B. Bosansky, C. Kiekintveld, V. Lisy, et al. Double-oracle Algorithm for Computing an Exact Nash Equilibrium in Zero-sum Extensive-form Games. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 335–342. 2013.

[15] V. Lisy, R. Pibil, J. Stiborek, et al. Game-theoretic approach to adversarial plan recognition. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, pp. 546–551. 2012. doi:10.3233/978-1-61499-098-7-546.

[16] P. Auer, N. Cesa-Bianchi, P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* **47**(2-3):235–256, 2002. doi:10.1023/A:1013689704352.

[17] C. B. Browne, E. Powley, D. Whitehouse, et al. A survey of monte carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on* **4**(1):1–43, 2012. doi:10.1109/TCIAIG.2012.2186810.

[18] M. Shafiei, N. Sturtevant, J. Schaeffer. Comparing UCT versus CFR in Simultaneous Games. In *IJCAI Workshop on General Game Playing*. 2009.

[19] S. Hart, A. Mas-Colell. *A Reinforcement Procedure Leading to Correlated Equilibrium*. Springer Berlin Heidelberg, 2001. doi:10.1007/978-3-662-04623-4_12.

[20] Agent Technology Center. *Computation game theory*, 2014. `http://agents.felk.cvut.cz/topics/Computational_game_theory` [2014-09-29].

[21] M. Lanctot. *Monte Carlo Sampling and Regret Minimization for Equilibrium Computation and Decision Making in Large Extensive-Form Games*. Ph.D. thesis, University of Alberta, 2013.

[22] M. Lanctot, R. Gibson, N. Burch, et al. No-Regret Learning in Extensive-Form Games with Imperfect Recall. In *ICML*, pp. 1–21. 2012. `arXiv:1205.0622v1`.