Some Variations of Upper Confidence Bound for General Game Playing

Iván Francisco-Valencia(✉), José Raymundo Marcial-Romero, and Rosa María Valdovinos-Rosas

Facultad de Ingeniería, Universidad Autónoma del Estado de México, Cerro de Coatepec S/N Ciudad Universitaria, 50100 Toluca, Estado de México, Mexico
if.valencia@hotmail.com, {jrmarcialr,rvaldovinosr}@uaemex.mx

Abstract. Monte Carlo Tree Search (MCTS) is the most used method in General Game Playing, area of the Artificial Intelligence, whose main goal is to develop agents capable of play any board game without preview knowledge. MCTS requires a tree which represents the states and moves of the board game which is visited and expanded using an iterations method. In order to visit the tree, MCTS requires a selection policy which determines which node is visited in each level. Nowdays, Upper Confidence Bound (UCB), is the most popular policy in MCTS due to its simplicity and efficiency. This policy was propose for the Multi-Armed Bandit Problem (MABP) which consists in set of slot machines each of which has a certain probability of give a reward. The goal is to maximize the accumulative reward that is obtained when a machine is played in a series of rounds. Other policy proposed for MCTS is Upper Confidence Bound$_{\sqrt{\cdot}}$ (UCB$_{\sqrt{\cdot}}$) whose goal is to identify the machine with the highest probability to give a reward. This paper shows a comparative between five modifications of UCB and one of UCB$_{\sqrt{\cdot}}$, this comparative has the goal of finding a policy which be able to identify the optimal machine as quickly as possible, this goal in MCTS is equals to identify the node with the highest probability to leading to a victory. The results show that some policies find the optimal machine before UCB, however, with 10,000 rounds UCB is the policy who plays the optimal machine more often.

Keywords: General game playing · Selection policy · Upper confidence bound

1 Introduction

For Bjornsson and Finnsson [5], General Game Playing (GGP) is the area of Artificial Intelligence of which the objective is to create intelligent agents who can learn, automatically, how to play a wide variety of board games, based only on the descriptions of the rules of the games. The foregoing implies that without prior knowledge about the game and while playing, the agent must be able to

© Springer Nature Switzerland AG 2019
J. A. Carrasco-Ochoa et al. (Eds.): MCPR 2019, LNCS 11524, pp. 68–79, 2019.
https://doi.org/10.1007/978-3-030-21077-9_7

develop strategies that allow it to win. Since its inception GGP makes use of methods based on MinMax or Alpha-Beta tree [11], this is due to the nature of boardgames, to which tree can be associated, where the root node represents the initial state of the game and each child node represents the status of the game after some movement has been made. Due to the above, the leaf nodes of the game tree correspond to the statuses where the game has ended, whereby the agent must only find a leaf node where he achieves a win; this is the reason for using methods based on search trees.

Monte Carlo Tree Search (MCTS) is the search method based on the most popular tree in GGP, as it has a better performance in game trees [6]. MCTS consists of four steps that are repeated cyclically, until a stop criterion is met: Selection, Expansion, Simulation, and Back Propagation. The stop criterion can be a limit number of simulations, execution time or number of iterations [6].

Selection In this step the method crosses the tree from the root node until it finds a node that still has children to add to the tree, once this node is found the Expansion step is reached. The route taken by this step is guided by a Selection Policy, which indicates which node should be explored in each level; an example of this policy is to choose the node that has the highest ratio between wins and visits.

Expansion In this step, a corresponding child node is added to the node found in the selection step.

Simulation Starting from the status represented by the newly added node, the method simulates playing the game by performing the movements of the players randomly until a result is obtained.

Back Propagation In this step, the result of the simulation step is propagated in all the nodes visited, updating the number of wins and the number of visits of each node.

Once the method ends, the movement that the agent must perform is chosen among the children of the root node which could be: the node with the highest number of wins, the node with the highest number of visits, the node that meets the two previous criteria, or the node is chosen based on the selection policy. MCTS has the advantage of being able to be used at any time during the game since the root of the tree can be any status of the game, another feature of MCTS is to be efficient by not having to completely expand the tree as it resorts to probability to choose the movement that has the highest chance of leading to a win, so it is also known as a probabilistic method.

In recent years efforts have been made to improve MCTS, mainly in the Simulation Step where attempts have been made for the simulation to reflect movements of real adversaries without being completely deterministic, highlighting the works of Cazenave [8–10], whose idea is to make use of online knowledge, by identifying the movements of previous iterations that led to wins to be used, with a greater probability, in future iterations.

Another step of MCTS where efforts are made to improve it is the Selection Step specific to the Selection Policy; in the beginning the average of wins of each

node was used as selection policy. However, new policies have been proposed, such as Upper Confidence Bound. In the Selection Step, at each level of the tree, MCTS has to make the following decision: Which node should be explored? The one in which the highest number of wins is obtained so far, or should we explore less promising nodes that may turn out to be better in future iterations? This decision is an instance of the Explore-Exploit Dilemma, which Auer et al. [3] describe as the search for a balance between exploring the environment to find profitable actions while taking the best empirical action as frequently as possible.

Another instance of the Explore-Exploit Dilemma is the Multi-Armed Bandit Problem (MABP), which consists in set of slot machines each of which has a certain probability of give a reward. The goal is to maximize the accumulative reward that is obtained when a machine is played in a series of rounds.

An algorithm that allows to decide which machine to activate in each round in MABP is known as Activation Policy, where Upper Confidence Bound (UCB) is the most popular, mainly because it is efficient, simple to implement and can be used at any time [3,4]. However, there are other policies that achieve performance close to UCB such as UCB2, ϵ-greedy, UCB-Tuned, UCB-Normal [3], UCB-Improved [4], UCB-V [2], UCB-Minimal [13] and Minimax Optimal Strategy in the Stochastic Case [1].

Because MABP and the decision made by MCTS in the Selection Step are instances of the Explore-Exploit Dilemma, it is possible to use Activation Policies as Selection Policies. In this case, each level of the tree is treated as a MABP where each node is equivalent to a slot machine; this idea was used for the first time by the agent Cadiaplayer, which made use of UCB as a selection policy, giving good results to such an extent that the combination of MCTS and UCB, known as Upper Confidence Bound Applied to Trees (UCT), became the state of the art of GGP.

The approach of activation policies like UCB and its similar ones is to minimize Cumulative Regret which is defined as the loss that is obtained due to the fact that the policy does not always choose the best machine [3]. However, this approach is not necessarily suitable for MCTS since in this the idea is to identify the node that is most likely to lead to a win. Another approach known as Simple Regret [7,14] has been proposed, and is more suited to MCTS [12], and which is defined as the difference between the expected reward of the optimal machine (the machine with the highest probability of giving a reward) and the expected reward of the machine that has been identified as the optimal machine, from this approach emerge $\text{UCB}_{\sqrt{\cdot}}$.

This paper presents a comparison between five modifications of UCB and one of $\text{UCB}_{\sqrt{\cdot}}$. In order to find a selection policy that is able to identify the machine as quickly as possible, the above would be equivalent in MCTS to identifying the node that has the highest chance of leading to a win at each level of the tree. The comparison was made in the regarding MABP in two scenarios: the first consists of the scenario proposed by Auer et al. [3], in the second one the use of the branching factor of the game tree of different board games is used to generate sets of machines where the proposed policies were tested. The results show that

certain policies find the optimal machine in the first iterations, although at close to 10,000 iterations it is UCB that activates the optimal machine.

2 Upper Confidence Bound

Auer et al. [3,4] formally define MABP by the random variables $X_{i,n} \in \{0,1\}$ with $1 \leq i \leq K$ and $n \geq 1$, where each i is the index of a slot machine, and K the machines available. By successively activating the i machine, the rewards $X_{i,1}, X_{i,2}, \cdots$ are obtained, which are independent and identically distributed according to an unknown law with unknown expectation μ_i.

UCB is the most widely used policy in MABP because it achieves logarithmic and uniform regret as n increases, and does not require information about probability distributions and is easy to implement.

UCB consists in the following:

1. Play each machine once.
2. Play the machine j that maximize $\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$
   where $\bar{x}_j$ is the average reward obtained by the j machine, n_j is the number of the times that the j machine has been played and n is the total number of plays done so far.
3. The previous step is repeated until a certain number of rounds is reached.

3 Upper Confidence Bound$_{\sqrt{\cdot}}$

Proposed by Tolpin and Shimony [14], the UCB$_{\sqrt{\cdot}}$ policy is the one used in MCTS and is focused on minimizing simple regret, and consists of:

1. Play each machine once
2. Play the machine j that maximize

$$\bar{x}_j + \sqrt{\frac{c\sqrt{n}}{n_j}} \tag{1}$$

   where $\bar{x}_j$ is the average reward obtained by the j machine, n_j is the number of the times that the j machine has been played and n is the total number plays done so far.
3. The previous step is repeated until a certain number of rounds is reached.

4 Policies Proposals

UCB is the most used policy in MABP and consequently in Monte Carlo Tree Search; in this section five modifications to this policy are presented:

$$UCB\text{-}A = \overline{x}_j + \sqrt{\frac{2 \log n}{n}} \tag{2}$$

$$UCB\text{-}B = \overline{x}_j + \sqrt{\frac{2\log n_j}{n_j}} \tag{3}$$

$$UCB\text{-}C = \overline{x}_j + \sqrt{\frac{2\log n_j}{n}} \tag{4}$$

$$UCB\text{-}D = \overline{x}_j \tag{5}$$

$$UCB\text{-}E = \overline{x}_j + \frac{n_j}{n} \tag{6}$$

The *UCB-A* policy makes use only of the total number of Machine Activations (number of simulations of the parent node in MCTS). The *UCB-B* policy makes use of the number of activations of the machine (number of simulations in the child node). The policy *UCB-C* is similar to UCB but with n_j y n exchanged. The *UCB-D* policy only takes the average of rewards obtained in the machine (the average of wins per node in MCTS) that means that this policy is only for exploitation. The *UCB-E* policy requires the average of the plays. Finally, *UCB-F* is a modification of the policy $\text{UCB}_{\sqrt{\cdot}}$ with n_j y n exchanged.

$$UCB\text{-}F = \overline{x}_j + \sqrt{\frac{2\sqrt{n_j}}{n}} \tag{7}$$

5 Comparative of Policies

In this section we compare the performance of the proposed policies with respect to UCB and $\text{UCB}_{\sqrt{\cdot}}$. Specifically, we can see how good the policies are in choosing the optimal machine. The choice to measure how much a policy chooses the optimal machine is because in the MCTS field it is equivalent to choosing the child node in which the highest number of wins is given.

The policies were compared in the MABP in two scenarios; the first is the one proposed by Auer et al. [3] and the second scenario is where the branching factor of a set of board games is used.

5.1 First Scenario

This scenario is the one proposed by Auer et al. [3] to prove the policies *UCB*, *UCB-T*, *UCB2*, *UCB-Normal* and ϵ-greedy. Auer et al. propose that the policies should be proven in 7 sets of machines, the Table 1 shows these sets with the probabilities of giving a reward of each of their machines.

For Auer et al. the sets A and D are easy to contrast because the reward of the optimal machine has low variance and the difference between the expected value of the optimal machine and suboptimal is wide. Sets C and G are hard sets because the reward of the optimal machine has high variance and the difference between the expected value of the optimal machine and suboptimal is small.

The policies were compared with the following conditions:

- Each of the sets proposed by Auer et al. were used.
- Each policy was tested 100 times in each set, from which the average of activation of the optimal machine was obtained.
- The policies were limited to 100,000 rounds.

Table 1. Sets of slots machine

Set	Probability of giving a reward									
A	0.9	0.6								
B	0.9	0.8								
C	0.55	0.45								
D	0.9	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
E	0.9	0.8	0.8	0.8	0.7	0.7	0.7	0.6	0.6	0.6
F	0.9	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
G	0.55	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45

Results. This Figs. 1, 2, 3 and 4 show the results obtained in each set and the Table 2 shows the average percentage of plays of the optimal machine. We can note at 100,000 rounds, *UCB* is the best policy because it has the best performance due it activates the optimal machine over 95.8% on average, the same happens at 10,000 rounds where the optimal machine is activated 80.6%. However, in lower rounds *UCB-A* and *UCB-B* are the policies that activate the optimal machine more frequently, over 71% at 1,000 rounds and over 57% at 100 rounds. It is worth highlighting that $UCB - B$ has performance similar to the performance of *UCB*, and it is the second-best performance at 100,000 and 10,000 rounds. The other policies have a performance bellow *UCB*, *UCB-A*, *UCB-B* and *UCB-D*, and it is *UCB-E* the worst policy due it only reaches 37% of the activation of the optimal machine. From the figures we can note that *UCB* in the first rounds it is dedicated to exploration in order to find the optimal machine without underestimate any other suboptimal machine, in these same rounds *UCB-A* and *UCB-D* are the polices that most quickly activate the optimal machine in all sets except for the set *C*. However, both policies tend to stagnate after 1,000 rounds and they do not overcome to *UCB*.

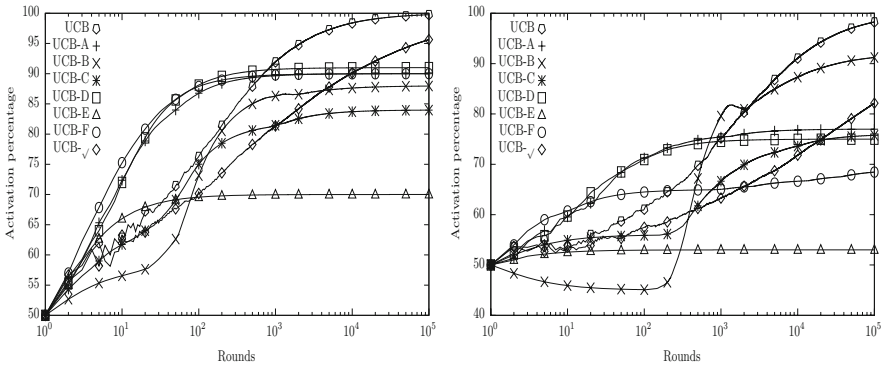

Fig. 1. Activations of optimal machine in sets *A* (left) and *B* (right)

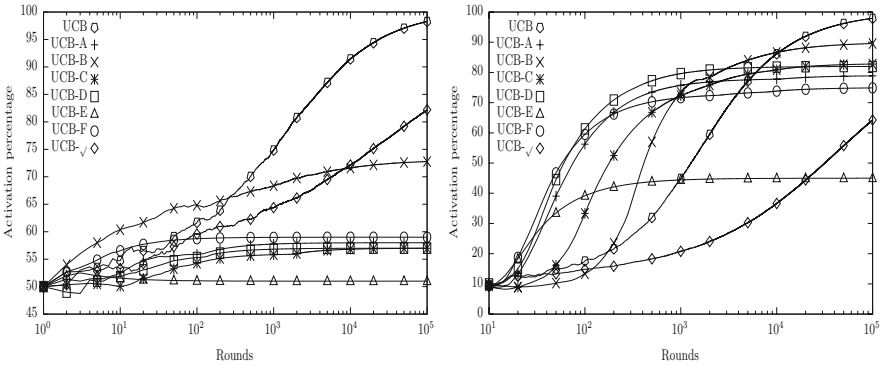

Fig. 2. Activations of optimal machine in sets C (left) and D (right)

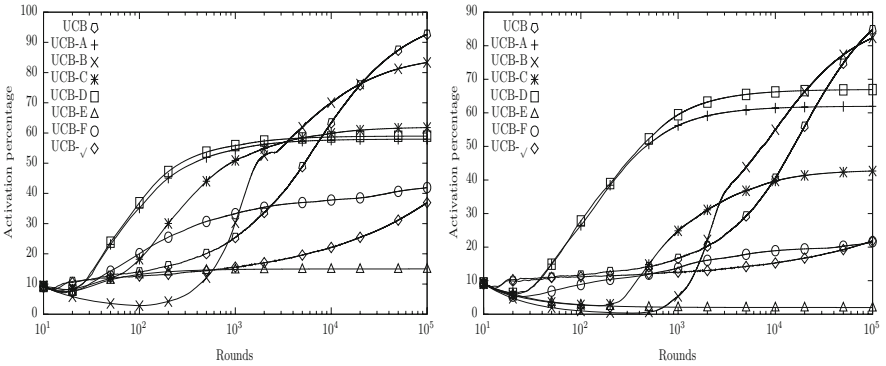

Fig. 3. Activations of optimal machine in sets E (left) and F (right)

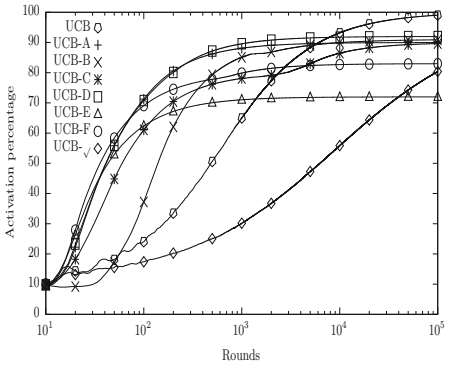

Fig. 4. Activations of optimal machine in set G

Table 2. Activation percentage of optimal machine

Rounds		UCB	$UCB-A$	$UCB-B$	$UCB-C$	$UCB-D$	$UCB-E$	$UCB_{\sqrt{\cdot}}$	$UCB-F$
100,000	$\overline{x}$	**95.8182**	73.3904	85.4395	70.5138	74.6906	37.5698	66.1962	62.6921
	σ	4.9840	13.0925	6.1034	15.8557	13.3010	28.8866	25.0811	22.2657
10,000	$\overline{x}$	**80.5987**	73.0574	78.3030	68.7177	74.4771	37.5555	51.9887	61.2636
	σ	19.4140	13.1107	12.1342	15.9161	13.3509	28.8716	26.0406	23.3561
1,000	$\overline{x}$	56.2810	71.0549	61.1777	61.4323	**72.3633**	37.4120	41.1454	58.9154
	σ	25.9311	13.8878	28.8332	18.1425	14.0022	28.7228	25.7844	24.7352
100	$\overline{x}$	37.9771	57.4729	33.8329	42.8214	**58.6314**	35.9771	34.7229	52.6000
	σ	25.1655	19.6814	26.9652	23.8225	19.3466	27.3797	24.2615	25.9624

5.2 Second Scenario

In this scenario the proposal policies were tested in the field of the MABP problem however we used the branching factor of the games that we show in the Table 3.

Table 3. Branching factor of games

Game	Branching factor	Game	Branching factor
Tic-Tac-Toe	4	Connect 4	4
Draughts (10×10)	4	Domineering (8×8)	8
Nine Men's Morris	10	Reversi	10
Fanorona	11	Lines of Action	29
Chess	35	Chinese chess	38
Japanese chess	40	Korean chess	92
Gomoku	210		

The branching factor is used because this will be the number of machines the polices will face if implemented in the Monte Carlo Tree Search

For this scenario the following conditions are required:

- The game branching factor is used.
- For each branching factor, five sets of machines with random probabilities were created.
- The policies were tested 100 times in 10,000 rounds in these sets, of which the average was obtained.
- We obtained the average of activations of the optimal machine.

Results. From the Table 4 we can note that when we have a branching factor under 11, it is *UCB-B* the policy with the best performance due it reaches between 74% and 83% of activations of the optimal machine, except when we have a branching factor of 4 where *UCB* is the best policy. For the rest of branching factors, we can note that *UCB-A* and *UCB-B* have the best performance due to these are the policies that activate the optimal machine more frequently. Form the Table 5 we can note that in all games the best policies are *UCB-A* and *UCB-B* given that in average they activate the optimal machine more frequently. From the Table 6 we can note that the behavior of the policies changes with *UCB-A* and *UCB-D* the best policies when we have a branching factor under 11. Surprisingly, in the Table 6, we can note for the branching factors 29, 35, 38 and 40, *UCB-F* has the best performance.

Table 4. Activation percentage of optimal machine in games at 10,000 rounds

BF		UCB	$UCB-A$	$UCB-B$	$UCB-C$	$UCB-D$	$UCB-E$	$UCB_{\surd}$	$UCB-F$
4	$\bar{x}$	**65.6611**	59.6806	57.9441	51.5995	56.9357	49.5891	50.3142	51.0588
	σ	15.3476	5.5503	11.2091	5.9047	11.1973	4.3158	11.6598	8.2216
8	$\bar{x}$	74.9371	75.1826	**82.8859**	74.4210	74.9500	52.5614	44.2984	63.2555
	σ	18.3563	16.9671	13.5736	14.5467	17.8437	16.2887	16.1516	19.9180
10	$\bar{x}$	67.8768	69.5031	**74.8487**	65.4245	69.7310	38.7614	33.7844	59.9355
	σ	13.2718	14.9729	15.1556	15.4648	16.5656	8.5786	7.7511	12.9619
11	$\bar{x}$	74.3563	77.8430	**80.1822**	73.1342	76.8665	34.9542	36.9537	65.4115
	σ	22.1472	24.4220	22.8700	23.0606	25.0331	17.8814	12.6490	28.0763
29	$\bar{x}$	37.2920	**76.8383**	55.7563	62.7048	74.1868	36.8889	10.1818	46.2529
	σ	11.0098	15.4155	29.1863	18.5392	14.9992	22.8625	1.4704	21.5763
35	$\bar{x}$	24.5105	**67.4117**	43.2063	53.0103	66.6836	29.6829	7.4822	40.0294
	σ	3.1729	21.0204	33.6807	28.1489	21.6409	23.6254	0.7376	29.2565
38	$\bar{x}$	22.7111	57.6530	30.0124	41.6675	**57.7975**	14.9365	7.0274	25.8340
	σ	3.9723	12.0977	25.3554	16.9108	13.0974	14.9539	0.5625	16.7117
40	$\bar{x}$	30.0559	72.9773	50.1235	59.0959	**73.6321**	26.2777	7.0432	34.8555
	σ	9.0977	19.1200	30.1434	25.1066	20.2007	29.5561	1.0251	31.7675
92	$\bar{x}$	6.3170	**36.9091**	7.3637	17.1300	33.3640	7.5018	2.0843	9.1248
	σ	0.9060	6.4753	14.5398	14.8989	8.0411	10.6270	0.0996	13.7220
210	$\bar{x}$	1.8606	45.1294	0.0100	1.5039	**45.8125**	0.9766	0.7544	0.3475
	σ	0.1406	18.3626	0.0000	2.0300	18.6298	1.9332	0.0139	0.5902

Table 5. Activation percentage of optimal machine in games at 1,000 rounds

BF		UCB	$UCB-A$	$UCB-B$	$UCB-C$	$UCB-D$	$UCB-E$	$UCB_{\surd}$	$UCB-F$
4	$\bar{x}$	52.2936	**59.2170**	54.3214	49.6422	56.3692	49.4910	42.3852	50.4402
	σ	13.0574	5.6605	10.1968	7.0983	11.2702	4.3013	7.1777	7.7528
8	$\bar{x}$	49.5066	72.8488	66.7468	66.7380	**72.9586**	52.2142	30.2922	59.5674
	σ	17.5118	17.3604	16.4454	15.3563	18.0711	16.1459	9.3635	19.8535
10	$\bar{x}$	38.6312	67.0240	54.7670	55.4538	**67.3690**	38.4144	22.1458	56.6776
	σ	9.4914	14.8311	10.2368	13.3721	16.0270	8.4904	3.4105	10.7863
11	$\bar{x}$	43.2396	**75.2802**	50.4530	62.3224	74.4496	34.5416	22.0318	59.9800
	σ	15.5835	24.9104	27.2729	23.6197	25.5042	17.6155	5.0195	27.1612
29	$\bar{x}$	11.9900	**66.7928**	35.8372	41.6526	65.1450	35.8894	6.4858	40.4814
	σ	2.1578	12.6583	27.1850	26.1605	11.9942	22.2133	0.4633	22.3180
35	$\bar{x}$	8.5186	**57.5004**	14.1398	28.7188	56.5842	28.6292	5.0170	35.1438
	σ	0.8929	16.3202	17.7661	20.9094	17.1572	22.8657	0.3096	28.2344
38	$\bar{x}$	8.0218	**47.9556**	12.4592	22.5314	47.5542	14.3646	4.6864	21.3432
	σ	0.7405	11.1791	23.7481	18.1138	11.8451	14.4796	0.2377	17.1685
40	$\bar{x}$	8.2068	63.0024	16.8792	25.9206	**63.4442**	25.1766	4.5156	30.2384
	σ	1.4715	17.5381	31.2955	32.3617	17.9930	28.4204	0.3370	31.4916
92	$\bar{x}$	2.2336	**21.1234**	0.1284	4.7348	21.0182	6.6180	1.5626	6.0978
	σ	0.1456	1.8250	0.0568	8.0212	4.5364	9.4044	0.0389	10.3251
210	$\bar{x}$	0.7472	9.4600	0.1000	0.3362	**9.6454**	0.7658	0.5976	0.2388
	σ	0.0323	9.6418	0.0000	0.3081	9.8170	1.3316	0.0017	0.2776

Table 6. Activation percentage of optimal machine in games at 100 rounds

BF		UCB	$UCB-A$	$UCB-B$	$UCB-C$	$UCB-D$	$UCB-E$	$UCB_{\surd}$	$UCB-F$
4	$\bar{x}$	39.2640	**55.5520**	47.0840	46.8160	53.2960	48.5100	35.7300	49.0280
	σ	6.3660	6.3454	10.9596	7.6774	10.9542	4.1571	3.9259	6.9773
8	$\bar{x}$	26.3080	57.5320	35.6840	44.2320	**58.5100**	48.7420	21.0560	52.1900
	σ	6.6413	17.6117	15.9314	13.6677	17.8331	14.7257	3.7780	16.2202
10	$\bar{x}$	18.9140	49.7680	17.6100	31.4280	**50.2840**	34.9440	15.7100	47.1020
	σ	2.5276	11.3634	6.3259	6.6079	11.6234	7.6148	1.2526	7.8605
11	$\bar{x}$	18.0800	55.9120	10.0180	23.9920	**56.6580**	30.4160	14.6320	45.0880
	σ	3.6091	20.0240	4.3396	12.7578	19.7044	14.9594	1.8329	18.9701
29	$\bar{x}$	5.3460	20.1220	16.8800	23.3360	19.4020	25.8940	4.6920	**27.5820**
	σ	0.3909	7.6491	13.6249	18.3071	7.6527	15.7421	0.2729	15.6954
35	$\bar{x}$	4.0440	16.0360	7.3080	11.5380	16.5060	18.0920	3.6740	**20.2440**
	σ	0.0739	9.3594	9.3838	12.6745	9.2657	15.3353	0.2782	18.6115
38	$\bar{x}$	3.7960	8.6560	6.7820	9.3140	8.8000	8.6460	3.0820	**10.1560**
	σ	0.0736	4.3635	11.5640	12.8030	4.8589	9.8454	0.0937	12.0551
40	$\bar{x}$	3.5040	11.7760	4.2520	11.4960	11.8560	14.1660	2.9740	**15.4960**
	σ	0.2938	8.9449	6.4641	17.5303	8.9641	17.1619	0.0372	19.4871

6 Conclusions and Future Work

From the first scenario we could note that *UCB* is the policy with the best performance due to it activates the optimal machine over 80% of the time after 10,000 rounds. *UCB-B* had a similar performance to *UCB* however it did not reach the percentage of *UCB*. In this scenario we could note that *UCB-A* and *UCB-B* are the policies that activate the optimal machine as quick as possible, however, in the last rounds they are outperformed by *UCB*. This behavior was repeated when we used set of machines based in branching factor of games and we could note that the performance of *UCB* decreased as the number of rounds increased, probably in late rounds *UCB* can outperformed the other polices.

Because *UCB-A* and *UCB-D* are policies that only use exploitation and due to the results that we got, we can conclude that when we have low number of rounds below 10,000, it is better to use exploitation polices but with a high number of round is better use *UCB*. However, we need to apply these policies in MCTS and GGP in order to get the real behavior. In both scenarios $UCB_{\sqrt{\cdot}}$ had the worst performance, this may be due to the wrong choice of the value of its constant, so we leave as future work to tune this value and compare its performance with the exploitation polices.

References

1. Audibert, J.Y., Bubeck, S.: Minimax policies for adversarial and stochastic bandits. In: COLT, pp. 217–226 (2009)
2. Audibert, J.-Y., Munos, R., Szepesvári, C.: Tuning bandit algorithms in stochastic environments. In: Hutter, M., Servedio, R.A., Takimoto, E. (eds.) ALT 2007. LNCS (LNAI), vol. 4754, pp. 150–165. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75225-7_15
3. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**(2–3), 235–256 (2002)
4. Auer, P., Ortner, R.: UCB revisited: improved regret bounds for the stochastic multi-armed bandit problem. Periodica Math. Hungarica **61**(1–2), 55–65 (2010)
5. Bjornsson, Y., Finnsson, H.: Cadiaplayer: a simulation-based general game player. IEEE Trans. Comput. Intell. AI Games **1**(1), 4–15 (2009)
6. Browne, C.B., et al.: A survey of Monte Carlo tree search methods. IEEE Trans. Comput. Intell. AI Games **4**(1), 1–43 (2012)
7. Carpentier, A., Valko, M.: Simple regret for infinitely many armed bandits. In: International Conference on Machine Learning, pp. 1133–1141 (2015)
8. Cazenave, T.: Playout policy adaptation for games. In: Plaat, A., van den Herik, J., Kosters, W. (eds.) ACG 2015. LNCS, vol. 9525, pp. 20–28. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27992-3_3
9. Cazenave, T.: Playout policy adaptation with move features. Theoret. Comput. Sci. **644**, 43–52 (2016)
10. Cazenave, T., Diemert, E.: Memorizing the playout policy. In: Cazenave, T., Winands, M.H.M., Saffidine, A. (eds.) CGW 2017. CCIS, vol. 818, pp. 96–107. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75931-9_7

11. Genesereth, M., Thielscher, M.: General Game Playing. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, San Rafael (2014)
12. Liu, Y.C., Tsuruoka, Y.: Modification of improved upper confidence bounds for regulating exploration in Monte-Carlo tree search. Theoret. Comput. Sci. **644**, 92–105 (2016)
13. Maes, F., Wehenkel, L., Ernst, D.: Automatic discovery of ranking formulas for playing with multi-armed bandits. In: Sanner, S., Hutter, M. (eds.) EWRL 2011. LNCS (LNAI), vol. 7188, pp. 5–17. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29946-9_5
14. Tolpin, D., Shimony, S.E.: MCTS based on simple regret. In: AAAI (2012)