2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)

Monte-Carlo Tree Search with Tree Shape Control

Oleksandr I. Marchenko, Oleksii O. Marchenko
Department of System Programming and Specialized Computer Systems
National Technical University of Ukraine "Igor Sikorsky Kyiv polytechnic institute"
Kyiv, Ukraine

***Abstract* — The paper, basing on analysis of the Monte-Carlo Tree Search (MCTS) method and specific features of its behavior for various cases of usage, proposes a new variant of the method, which was called as Monte-Carlo Tree Search with Tree Shape Control (MCTS-TSC) and which uses original Depth-Width Criteria (DWCs) for both tree shape estimation and control during search and for estimation and selection of potentially better options for search continuation. Proposed Tree Shape Control (TSC) technique can be used with some other tuning, pruning, and learning techniques. Besides, it can provide better scheduling of MCTS parallelization.**

Keywords — Monte-Carlo Tree Search (MCTS); game tree; MCTS improvement techniques; exploitation-exploration dilemma; tree shape control; MCTS parallelization

I. INTRODUCTION

Monte-Carlo Tree Search (MCTS) is already well-known method for representation of a game information in tree form and for efficient searching of the most reasonable moves for the game continuation using this tree. To present day, many implementations of MCTS were proposed along with many techniques for its improvement [1]. Games have very different branching factor of their game trees, e.g. it is 35 for the game Chess while the game Go has dramatically greater branching factor 250 [2]. It is an unrealistic task to build the complete tree for a game with so large branching factor like Go has. Thus, on the one hand, the greater number of tree branches are already used in a tree, the better analysis for selection next move can be done; on the other hand, building a tree with many branches at each node requires too much time and memory space. This follows that at least the next tasks should be solved efficiently:

- The task of selection the best moves at each node of a search tree using theoretical improvements of the MCTS method. This task includes sub-tasks such as solving known dilemma between amount of exploitation and exploration performed, correct pruning of definitely bad moves, avoiding selection of a bad move which looks like as the best move at the moment (traps), etc.
- The task of search process speedup by parallelization techniques (hardware independent parallelization).
- The task of search process speedup by effective usage of high-performance hardware (hardware dependent parallelization).

Good solution of the first task is crucial to get better next move each time and a strong player in result.

Two latter tasks are very important as well because in general case there is no guarantee that the best move would be selected in appropriate time without having appropriate speed of search. Solving the first task many various techniques were used trying to perform selection of the best moves among sets of non-visited yet and sub-optimal nodes, e.g. Accelerated UCT [3], BAST [4], FPU-UCT [5], UCT with pruning ideas [5], UCT-RAVE [6], and some others [1]. A common feature of these techniques is collecting some additional information basing on which such efficient selection of moves could be performed. Some of MCTS improving techniques use tree shape related criteria, in particular depth, branching factor, and some others, for estimations. Importance of controlling these criteria is discussed among other points in [3, 7, 8, 9, 10], in particular it was noticed that different games or search techniques might build different kinds of a search tree shape. There were got both positive and negative results for some aspects of game behavior and tree shape mutual influence, but it was noted that in any case "further work in this area could bear fruit" [9].

So, we think that dependency between shape of a search tree and efficiency of MCTS is worth to be investigated deeper. Some approaches for solving this topic and exploitation-exploration dilemma along with other theoretical improvements are discussed in Sections III, IV. The original Depth-Width Criteria (DWCs) and a new variant of MCTS, which called Monte-Carlo Tree Search with Tree Shape Control (MCTS-TSC), are proposed in Section V. These criteria, based on the depth-width approach [11], allow game tree shape estimation and control during searching process with the aim of selection potentially better options for search continuation. Another direction of the criteria usage is organization of better parallelization of the search process and better utilizing of hardware resources. This direction is discussed briefly in Section VI.

II. GENERAL MCTS APPROACH

A. MCTS Steps

Each node of a MCTS search tree can be considered as a root to a game position and contains information about number of wins played from this node and total number of playouts done through it. Usually, these numbers are written in a node by the slash symbol (number of wins / total number of playouts). Figure 1 shows an example of a MCTS tree, nodes of which contain results of some playouts.

978-1-5090-3006-4/17/$31.00 ©2017 IEEE

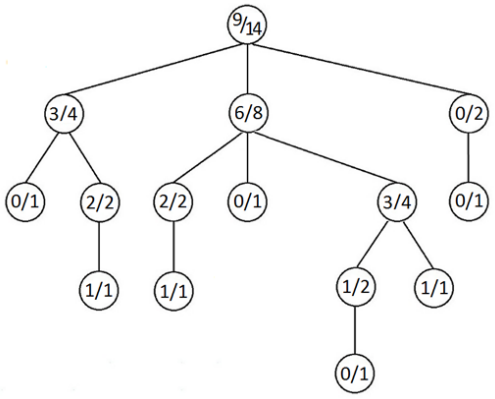

Fig. 1. MCTS search tree

General schema of MCTS method consists of the next four steps [1]: selection, expansion, simulation, backpropagation.

At the selection step each next node in a search tree is selected according to a special policy which is called Tree Policy. One of the most popular and effective tree policies is the policy of using Upper Confidence Bounds (UCB) method. According to a variant of UCB, which is called UCB1 [1], a node with the maximal value of the formula shown in Figure 2 is taken at the selection step.

$$UCB1 = \frac{NodeWins}{NodePlayouts} + 2 * C_{UCB1} * \sqrt{2 * \frac{\ln(ParentPlayouts)}{NodePlayouts}}$$

Fig. 2. UCB1 formula

This formula consists of two terms. The meaning of the first term is reward of a node (move) and the second term means approximation of upper bound at the confidence interval. In other words, maximization of UCB1 value allows selecting a node which possibly can be the best in future. C_{UCB1} parameter serves for tuning search process. Usage of UCB1 formula for MCTS was proposed by L. Kocsis and C. Szepesvári in [12], they called it Upper Confidence bounds applied to Trees (UCT).

III. MCTS Challenges

A. Balancing Exploitation with Exploration (Exploitation-Exploration dilemma)

The main goal of a Tree Policy is to decrease time spent for analysis of weak moves. It can be achieved by balancing exploitation with exploration that forms so-called exploitation-exploration dilemma. Exploitation is the use of currently the best nodes (moves), which are already presented in a tree, at each level during the descent at the selection step, while exploration is investigation of yet moves which are not the best for the moment but which can potentially become the best in result.

Various techniques were proposed as Tree Policies for solving exploitation-exploration dilemma [1, 12], such as UCB1 tuning, collecting additional information in nodes for better analysis, bias and heuristics techniques, machine learning methods based on domain specific information.

C_{UCB1} parameter (Figure 2) is taken so that to optimize balance between exploitation and exploration: greater value of C_{UCB1} corresponds to more extensive exploration. Raghuram Ramanujan and Bart Selman discussed exploitation-exploration dilemma in [8] among other points and influence of the C_{UCB1} parameter on it in particular. They detected that "for small values of C parameter, UCT misses too many moves and does poorly. With too large C, UCT is not sufficiently focused and the action utilities do not converge quickly enough to produce competent play". (They denoted C_{UCB1} as C.)

UCB1 formula focuses search process automatically on the most promising currently variants of further moves simultaneously with enabling enough exploration to find potentially better moves which are currently absent in the tree. We consider our proposed below Tree Shape Control (TSC) technique in some sense as a modifier for better exploitation-exploration decisions.

B. Traps and Optimistic Moves

Game traps [8] and so-called optimistic moves [7] are other MCTS challenges. "Optimistic moves is a name we have given moves that achieve very good result for its player assuming that the opponent does not realize that this seemingly excellent move can be refuted right away" [7]. The same thought is noticed in [4], they "show that UCT is "over-optimistic" in some sense, leading to a worst-case regret that may be very poor". It is concluded in [7] that MCTS could behave much better in case of "pruning these optimistic moves early on". Besides, [3] noted that "UCT's playouts often mistakenly evaluate winning positions as losses or vice versa" and "UCT may su□er from deceptive structures in the currently built tree" which "tends to appear when MCTS must select the best move at a node with only a few promising children".

Good decisions for these cases are based on the proper estimations of all sub-optimal moves at a node to ensure selection of really the best move. It is obvious that to take such decisions these sub-optimal moves should be investigated better that (in its turn) can be done by improving exploration of these moves. So, exploring other branches with about the same values is very important because it can turn out that one of these branches is the best indeed. Related point is better exploring of non-visited nodes because the best move (node) can be just among them. Some decisions for this issue, FPU (First Play Urgency) and UIP (Use Information of Parents) were considered in [5]. These techniques are modifiers for better exploring of non-visited nodes and can improve exploration process. But it should be noted that FPU definitely requires greater computation power or budget time to explore all non-visited yet nodes with first-play urgency.

We consider that in attempting to get better exploitation-exploration correspondence and better exploring of both existing sub-optimal and non-visited yet nodes, along with avoiding deceptive structures, noticed in [3], we should try to increase number of promising children at a node and improve their estimation. This is just the thing that could be potentially reached by the proposed TSC approach (Section V).

IV. Consideration of Tree Shape in MCTS

Different games might have different typical shape of trees. E.g. the greater branching factor of a game (corresponds to potential width of a game tree), the more complex (and maybe interesting) this game is. From the other side, simple games most likely will have narrow skeleton-like trees. Besides, H.Finnsson and Y.Björnsson [7] noted two significant statements that "relative to the number of nodes in the trees the depth and width can be varied, allowing us to answer the question if MCTS favors one over the other. …We found that it depends on the game itself whether MCTS prefers deep trees, big branching factor, or a balance between the two. Apparently small nuances in game rules and scoring systems, may alter the preferred game-tree structure" and MCTS's strength "appears to be dependent on the rules of the game being played".

So, we think that if we know typical shape of trees for a particular game then, using this information, we could help MCTS in correction its searching process directing this process to build just typical for this game shape of its tree.

A. Tree Shape Control by UCB1

Solving exploitation-exploration dilemma by UCB1 formula has the aim of increasing MCTS search process efficiency and in fact leads to controlling search tree shape as side effect. C_{UCB1} parameter is one of techniques for controlling exploitation-exploration ratio.

It was determined that the greater C_{UCB1} value, the greater number of explorations will be performed, and in result shape of a tree will increase in width [1]. In fact, C_{UCB1} is a parameter which allows controlling MCTS tree shape. If C_{UCB1} has lesser value, then the deeper and narrower tree will be built, while if C_{UCB1} has greater value, then the tree will be wider [13]. Another interesting result was got in [8]. For the domain, which was investigated (the game of Mancala), authors were able to "obtain a tree that strikes a balance between the two extremes; it is dense at shallow depths, but more sparse and focused at deeper levels". That happened with the average value of the C_{UCB1} parameter (it is denoted as C in their paper).

B. Researches on Relationship Between Tree Shape and MCTS or Games

Unfortunately, for now number of researches, which touch relationships between search tree shape and its impact on MCTS search effectiveness and quality of games, is not great. Let's make a short review of works in [3, 7, 8, 9, 10].

In [3] a special parameter λ is used as a parameter of decay ranging (0,1] for tuning of UCB1 formula, and an important conclusion regarding shape of trees is made basing on it: "it would be necessary for Accelerated UCT to automatically change the value of λ, based on a few factors such as the shape of the current search tree". This conclusion is very interesting for us because we think that using λ together with our TSC technique (Section V) could be profitable.

Some ideas described in [7, 8] about links between tree shape and MCTS run were noted above. Another important issue for our research is "balance between the tree depth and the branching factor" [7]. Using custom-made games, it was demonstrated that in the situation when number of good moves for selection in a node is growing up then probability of picking a wrong move increases as well. This means when there are many good variants to choose, MCTS might take a wrong decision for selection of the best one.

Reference [9], using many estimation criteria (tree features) and in particular some tree shape related criteria such as "variance in average width", "variance in maximum depth", "average branching factor", etc., investigated two hypotheses: (1) whether a machine learning system, trained on UCT trees for a selection of games, could accurately predict the quality of a new game it was presented with and (2) whether a machine learner could be found which could distinguish playable games from unplayable ones. Results for the second one were positive while result for the first hypothesis was it is unlikely that game quality could be accurately predicted basing on a tree features. As conclusion, it was noted that availability of more data would "provide a more definitive evaluation of the hypothesis" and "undoubtedly increase the chance of the machine learning systems finding a link between UCT tree shape and playability".

Reference [10] is not related just to research of search tree shapes impact because it is devoted to applying and tuning MCTS to the game Arimaa in general, but some analysis and thoughts on this topic are made. The most important point for us in this research is that it uses "the favorable shape of the tree" for making decisions. Besides, tuning of UCB1 formula, some additional techniques for controlling tree shape like FPU [5] and a variation of standard trick to limit the size of the tree (maturity threshold) were investigated in this work as well.

As the conclusion for Section IV it might be said that researches in [3, 7, 8, 9, 10] use shape features of a search tree in some forms as criteria and they got interesting results, but these investigations are directed to different topics than we have as an aim to investigate in our research.

V. MCTS with Tree Shape Control (MCTS-TSC)

The initial idea of our research is the following. If we know that a game or an application "likes" (i.e. has "small nuances in game rules and scoring systems" which "may alter the preferred game-tree structure" [7]) wider search tree than MCTS currently builds for it, then we can try to tune search process by some technique to force MCTS for building wider tree and vice versa. Information for such tuning might be collected by various already known techniques (e.g. basing on dependency on branching factor and on some other possible criteria) or by new techniques including some machine learning methods. This way we can help MCTS to build search trees which would correspond to stronger players.

As a rule, investigators of MCTS try to improve UCB1 formula for MCTS mostly from the "internal" point of view on a game tree (by "internal" criteria), i.e. taking into account various cases of wins/losses, heuristics based on domain-dependent knowledge, specific cases appeared during a game, etc., though some "external" criteria that reflect such "external" representation of a tree as its branching factor, depth, and some others, are used too (see Section IV). However, e.g. branching factor, used in these investigations to reflect tree width, is a

good criterion only for estimation of potential width of a tree which can be built in general, but it is not enough good to estimate current shape of a sub-tree of the already built tree, because branching factor indicates only local width of a node but not width of its actual sub-tree.

A. Depth-Width Criteria for Tree Shape Control

For solving some issues described above we propose a new Tree Shape Control (TSC) technique to improve UCB1 formula for MCTS from the "external" tree shape point of view to make MCTS exploitation-exploration decisions better. According to this idea, MCTS variant with Tree Shape Control was called MCTS-TSC. This technique uses two original Depth-Width Criteria (DWCs), DWC1 and DWC2, to estimate and control the actual shape of all sub-trees of a search tree.

DWCs are formulas for estimation of the current shape of a search tree, and are based on the depth-width ratio. DWCs are calculated for each node basing on two properties of tree shape, depth (NodeDepth) and width (NodeWidth):

- All terminal nodes (leaves) have both values of NodeDepth and NodeWidth equal to 1.
- NodeDepth of a non-terminal node is calculated as the maximal value of its children depths plus 1.
- NodeWidth of a non-terminal node is calculated as sum of its children widths.

Calculating process of NodeDepth and NodeWidth values for nodes during building a tree is shown in Figure 3.

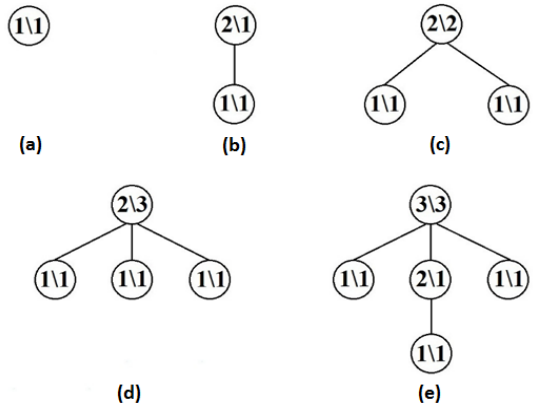

Fig. 3. NodeDepth and NodeWidth calculation

The first number is NodeDepth and the second one is NodeWidth of a sub-tree. To distinguish these values from the NodeWins and NodePlayouts pair of values in nodes we use back-slash symbol "\" instead of slash symbol "/". In other words, 6/8 means NodeWins/NodePlayouts and 6\8 means NodeDepth\NodeWidth. The same tree as in Figure 1, but with NodeDepth and NodeWidth calculated is shown in Figure 4.

Resulting values of the tree shape criteria DWC1 and DWC2 are obtained according to the next formulas.

$$DWC1 = C_{DWC1} * NodeDepth / NodeWidth$$

$$DWC2 = C_{DWC2} * (P * NodeDepth - NodeWidth) / NodeWidth$$

where C_{DWC1} and C_{DWC2} are parameters (similar to C_{UCB1} in UCB1 formula) to control DWCs impact onto MCTS search process. Meaning of the parameter P is discussed below.

DWCs are used as additional summands to the UCB1 formula. They can be used for a particular game either individually or together in a tree. We propose the following variants of the tree shape control (TSC) technique for MCTS:

$$TSC1 = UCB1 + DWC1$$

$$TSC2 = UCB1 + DWC2$$

$$TSC3 = UCB1 + DWC1 + DWC2$$

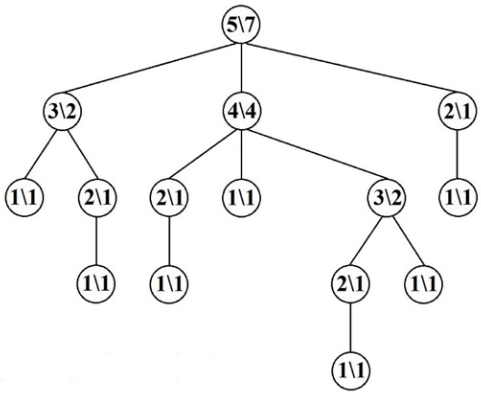

Fig. 4. NodeDepth \ NodeWidth calculation for a tree

One more variant of tree shape control (let's name it TSC4) can be proposed with using NodeDepth/NodeWidth ratio as criterion for dynamic tuning of standard UCB1 parameter C_{UCB1} by some technique for each node.

B. MCTS-TSC Discussion

Zero-value of C_{DWC1} and C_{DWC2} corresponds to no impact of DWCs on MCTS regular run.

TSC1 controls shape of a building tree adding the extra number DWC1 to UCB1 formula for nodes. The greater values of NodeDepth/NodeWidth ratio and C_{DWC1}, the greater impact of TSC1. If C_{DWC1} has positive value, then the impact will be in direction to better investigation of narrower sub-trees and in result to build wider sub-trees. And vice-versa, negative value of C_{DWC1} will force building deeper sub-trees because DWC1 is subtracted from the current value of UCB1. It is similar to impact of increasing/decreasing C_{UCB1} parameter, but TSC1 works depending on the shape of a tree as an additional factor.

If it is more convenient for a particular game/task, the opposite NodeWidth/NodeDepth ration can be used as well.

An example in Figure 5 shows how TSC1 (based on DWC1) switches selecting tree branches.

If standard UCB1 formula selects branch #1 then TSC1 technique changes this decision to another sub-optimal branch #3 with narrower shape.

It should be noted that NodeDepth/NodeWidth ratio cannot be negative, so it makes a bias to more or less wide sub-trees

(depending on sign of C_{DWC1}) at a node always, i.e. regardless of what is greater for the sub-tree, depth or width.

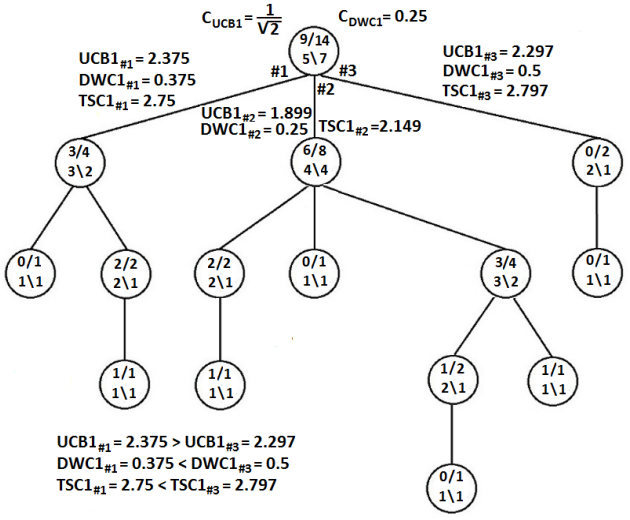

Fig. 5. An example how TSC technique works

We consider it is important to use TSC technique based on DWCs with some other techniques, which are based e.g. on such criteria as effective branching factor [9] or on a variant of FPU approach and which provide widening of the search tree in cases when DWCs show necessity to increase just width of a sub-tree or to build just narrower sub-tree.

DWC2 criterion allows to estimate and control shape of a tree in the manner of negative feedback. Zero-level for negative feedback is regulated by P parameter. If a sub-tree shape becomes deeper than it is set by P parameter for the difference between NodeDepth and NodeWidth, then (*P* * *NodeDepth* – *NodeWidth*) has positive value and DWC2 has positive value as well. In result, MCTS will be directed to build wider tree. If a sub-tree shape becomes wider than it is set for it by P parameter, then DWC2 has negative value and the situation is the opposite, i.e. MCTS search process will adjust the sub-tree to be narrower.

TSC2 and TSC3 work in a tree in a similar way as TSC1.

Both C_{DWC1} and C_{DWC2} parameters can be set as the same fixed constants for all nodes or as different constants for nodes at different depth levels (progressive values) or as variables depending on some heuristics and domain-dependent knowledge or learned by a machine learning technique. In the most general case each node can have its own values for these parameters tuned by learning techniques for building different parts of a search tree with different shape. E.g. using TSC we could get various tree shape effects like the effect in [8], i.e. a tree which is "dense at shallow depths, but more sparse and focused at deeper levels". It is important that these effects could be achieved in more flexible and controlled way. I particular, C_{DWC1} and C_{DWC2} parameters could be set with greater values for nodes with many non-visited yet and/or seldom visited children to have wider shape of a search tree with better exploration.

At the same time, using TSC technique for tuning of UCB1 formula does not exclude simultaneous utilizing of other techniques and most likely, the better usage of the TSC will be just together with some tuning, pruning and learning techniques including tree shape related ones as well. We believe that combination of the TSC technique with some biased [6], C_{UCB1}-tuned [8], and pruning techniques looks to be promising, because TSC allows achieving better exploration of non-visited and little-visited nodes, so that biased and pruning techniques would have more information for better defining of bias and efficient pruning of definitely bad branches. Besides, we think that DWCs criteria could be combined efficiently with usage of such tree shape related criterion as "effective branching factor" [9] or as parameter λ [3].

Moreover, we suppose that joint usage of TSC together with techniques based on domain-dependent knowledge or with machine-learning techniques can strengthen these techniques and produce good results as well. E.g. DWCs could be used as an additional term in *score(d)* formula proposed by Guillaume Chaslot at al. in [14] for combining o□ine, transient, online learnings and expert knowledge, with an exploration term.

All these variants of MCTS tuning are just subjects for further research. Because different games could "like" different shape of trees (Section V) then it is obviously expected that different TSC tuning techniques using DWCs can be efficient for different games and domains.

VI. Using TSC for Better Parallelization of MCTS

As it was discussed above, increasing amount of exploration is very advisable. Naturally, additional exploration requires additional time, so parallelization of MCTS and exploration process in particular has critical importance. If we are able to increase number of explorations by parallelization for no cost for the total time of MCTS search, then it would be a great solution for the case. We consider that proposed TSC techniques, in particular DWCs criteria, can be used not only for the cases discussed earlier but for better scheduling of parallelization as well.

Let's come back to the issues of non-visited and sub-optimal nodes along with mistaken (traps) and over-optimistic moves [4, 5, 7] but now from parallelization point of view. Exploring other branches with about the same good values at a given node (i.e. sub-optimal nodes) helps to identify the really best branch which was not clear before. So, it is very important if such additional exploration can be got for "no cost" during the same time by use of currently free hardware resources, and the task is how TSC can help to schedule properly these free hardware resources just for additional exploration.

DWCs, in particular NodeWidth, in difference to branching factor, provides information about width of a sub-tree not only for a particular node (that equals to branching factor), but about width of the whole sub-tree from this node. Such information, along with depth of the sub-tree, allows better defining of amount and kinds of available hardware resources, which will be required for parallelization of the all branches of the sub-tree (in width and depth), and schedule them in a proper way. Thus, due to TSC technique, it is possible to schedule extra exploration of the most promising branches of a game tree by use of currently free hardware resources without slowdown of

running the other parts of any MCTS variants, because these extra explorations would be done in parallel.

Of course, if MCTS is run on a computer system with few computers/processors/cores, then possibilities for additional exploration in parallel are limited. But, with increasing of number and productivity of computer system units, possibilities for the search parallelization grow as well. Though, from other side, complexity of effective parallelization implementation increases in this case, too.

Ideas, discussed in this Section, have not been implemented yet. So, after realization and investigation, some corrections and adjustments are possible.

VII. CONCLUSION

Basing on analysis of MCTS properties and already known techniques for control of the search tree building process, the original Depth-Width Criteria (DWCs), which are calculated by some rules, along with Tree Shape Control (TSC) technique were proposed in the paper. Let's note advantages that can be got by using TSC technique for running MCTS.

TSC technique can be used, e.g. if it is known in advance, what shape of search tree (narrow, "skeleton-like", wide, very wide, shallow, deep, very deep, etc.) is typical for efficient search of proper moves for a particular game or domain, or such knowledge can be got by some machine learning technique. If this information is known for particular tasks, then, taking it into account, a new possibility for dynamic control of search tree shape appeared, which allows to guide further tree building process in the required direction.

All variants of TSC technique, considering specific features of concrete games/domains and typical for them shape of trees, afford ground for more flexible distribution of time between exploitation and exploration. It is very important that DWCs can be used as additional terms to UCB1 equation and it is promising to utilize them as parameters for MCTS implementations with usage of neural networks, genetic algorithms, reinforcement learning, etc.

TSC technique allows both defining sub-trees, shape of which should be corrected to find new promising nodes for a game continuation, and better allocating available hardware resources for search parallelization in a given sub-tree.

Because calculation of DWCs is simple therefore usage of DWCs for tree shape control will not have significant impact on MCTS performance.

Detalization of applying the proposed TSC technique in various cases and creation of parallelization techniques on its base, can be recommended as directions for further investigations, and in particular, the following topics:

- Effectiveness of applying MCTS-TSC to various kinds of games and domains.
- Usage of TSC together with other exploitation/exploration improving techniques.
- Usage of TSC in various machine learning techniques.
- Usage of TSC for tuning parallelization process for various multicore, multiprocessor and multicomputer systems which include both CPUs and GPUs.
- It is supposed that DWCs criteria could be used to formulate a generalized model for MCTS dynamic parallelization with the goal to create faster MCTS implementations for concrete applications.

Generally, the proposed technique for controlling shape of a search tree can be applied not only for MCTS but in other domains where it might be useful as well.

REFERENCES

[1] Cameron Browne, Edward Powley, Daniel Whitehouse, et al. A Survey of Monte Carlo Tree Search Methods. IEEE Trans. on Computational Intelligence and AI in Games. - Vol. 4. - No. 1. - March 2012, pp.1-48.

[2] Alan Levinovitz. The Mystery of Go, the Ancient Game That Computers Still Can't Win, access https://www.wired.com/2014/05/the-world-of-computer-go/

[3] Junichi Hashimoto, Akihiro Kishimoto, Kazuki Yoshizoe, and Kokolo Ikeda. Accelerated UCT and Its Application to Two-Player Games, Advances in Computer Games, 13th International Conference, ACG 2011, Tilburg, The Netherlands, November 20-22, 2011, Revised Selected Papers, pp.1-12.

[4] P.-A. Coquelin and R. Munos. Bandit algorithms for tree search. In Proc. of the 23rd Conference on Uncertainty in Artificial Intelligence, (UAI2007), pages 67–74. AUAI press, 2007.

[5] Yizao Wang, Sylvain Gelly. Modifications of UCT and sequence-like simulations for Monte-Carlo Go. IEEE Symposium on Computational Intelligence and Games, CIG 2007, 1-5 April 2007.

[6] Sylvain Gelly, David Silver. Achieving Master Level Play in 9 x 9 Computer Go. Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, 2008, pp.1537-1540.

[7] Hilmar Finnsson and Yngvi Björnsson. Game-Tree Properties and MCTS Performance. GIGA 2011: Proceedings of the 2nd International General Game Playing Workshop, 2011, pp.23-30.

[8] R. Ramanujan and B. Selman. Trade-offs in sampling-based adversarial planning. Proc. 21st Int. Conf. Automat. Plan. Sched., Freiburg, Germany, 2011, pp.202-209.

[9] G. M. J. Williams. Determining Game Quality Through UCT Tree Shape Analysis. M.S. thesis, Imperial Coll., London, 2010

[10] Tomáš Kozelek. Methods of MCTS and the game Arimaa. M.S. thesis, Charles University in Prague, 2009.

[11] O.O.Marchenko, O.I.Marchenko. "Depth-Width" Criterion for Control of the Search Tree Shape Using Monte-Carlo Method. Komp'uterno-integrovani tekhnologii: osvita, nauka, vyrobnytstvo, No.24-25, 2016, pp.42-47 (in Ukrainian).

[12] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo Planning, in Euro. Conf. Mach. Learn. Berlin, Germany: Springer, 2006, pp.282–293.

[13] Joel Veness, Kee Siong Ng, Marcus Hutter, William Uther, David Silver. A Monte-Carlo AIXI Approximation. Journal of Artificial Intelligence Research 40 (2011), pp.95-142.

[14] G. M. J.-B. Chaslot, C. Fiter, J.-B. Hoock, A. Rimmel, and O. Teytaud. Adding Expert Knowledge and Exploration in Monte-Carlo Tree Search, in Proc. Adv. Comput. Games, LNCS 6048, vol. 6048, Pamplona, Spain, 2010, pp. 1–13.