Journal of Computer Science and Information Technology
June 2021, Vol. 9, No. 1, pp. 13-22
ISSN: 2334-2366 (Print), 2334-2374 (Online)

Published by American Research Institute for Policy Development
DOI: 10.15640/jcsit.v9n1a2
URL: https://doi.org/10.15640/jcsit.v9n1a2

Using Counterfactual Regret Minimization and Monte Carlo Tree Search for Cybersecurity Threats

Nii Emil Alexander Reindorf[1] & Paul Cotae[2]

Abstract

Mitigating cyber threats requires adequate understanding of the attacker characteristics in particular their patterns. Such knowledge is essential in addressing the defensive measures that mitigate the attack. If the attacker enters a network system, the game tree that modelsthose resources can generate a counter to such threats. This is done by altering the parity in the next game tree iteration which yield an adequate response to counter it. If an attacker enters a network system, and a game tree models the resources he must interface with, then that game tree can be altered, by changing the parity on the next to last iteration. This paper analyzes the sequence of patterns based on incoming attacks. The detection of attacker's pattern and subsequent changes in iterations to counter threats can be viewed as adequate resource or know how in cyber threat mitigations. It was realized that changing the game tree of the hacker deprives the attacker of network resources and hence would represent a defensive measure against the attack; that is changing varying or understanding attacker paths, creates an effective defensive measure to protect the system against the incoming threats. In this paper we analyze a unique combination of CFR and MCTS that attempts to detect the behavior of a hacker. Counterfactual Regret (CFR) is a game theory concept that helps identify patterns of attacks. The pattern recognition concept of Monte Carlo Tree Search (MCTS) is used in harmony with CFR in order to enhance the detection of attacks.

Keywords—Incomplete information games, extensive form games, counterfactual regret minimization, CFR, MCTS, Nash equilibrium

Introduction

A fundamental concept in cybersecurity is reducing data loss among other resource loss to malicious individuals. As such, mitigating such threats is essential in among other issues such as preventing network penetration and recognizing illicit activity. However, the nature of threats keeps on changing and hence determining patterns used by hackers could be essential in detecting incoming cyberattacks. These cyberattacks are perfect examples of imperfect information games. An example of an imperfect information game is poker. Poker is said to be imperfect because the strategy of the players is unknown. An example of a perfect information game would be chess because all the moves for all the chess pieces are known. For the purpose of this paper, the cybersecurity game of attacker versus administrator is modeled as an imperfect information game because strategies of the players are unknown.

CFR consists of the family of algorithms which converges to Nash equilibrium in zero-sum games. While the calculations for the CFR can be cumbersome, the main idea involves creating better futures through learning from past mistakes. In CFR, there is an attempt to minimize regrettable decisions by taking the opposite of the bad decision, iteration after iteration. In other words, we loop through a scenario with decisions and counter decisions of a game tree and then prune the tree so that the bad decisions or regret are minimized through changes in the game tree iterations. CFR has an advantage that the memory required is linear in the size of the information set of the games. For each play in performing CFR, a separate information set tree is implemented.

[1] Computer Science and IT Department, University of the District of Columbia, Washington, DC 20008, USA ,niiemil.alexanderrei@udc.edu
[2] Electrical and Computer Engineering Department, University of the District of Columbia, Washington, DC 20008, USA, pcotae@udc.edu

Among the most critical and well-studied topics in artificial intelligence are planning issues. Tree search algorithms that simulate ahead into the future, evaluate future states, and back up those assessments to the root of a search tree most usually solve them. Monte-Carlo Tree Search (MCTS) is one of the most common, efficient and commonly used among these algorithms. A traditional MCTS implementation uses cleverly built rules that are tailored for the domain's unique characteristics.

Such rules govern where the simulation goes through, what to measure in the states that are entered, and how to back up those assessments. Usually, MCTS is regarded as an online planner, where a decision tree is constructed as the root node starts from the current state. The standard purpose of MCTS is to propose an intervention for only the root node. The system moves forward after the action is taken, and a new tree is created from the next state (statistics from the old tree may be partially saved or completely discarded). Thus, MCTS is a "local" procedure (in that it only returns an action for a given state) and is fundamentally different from the approaches to value function approximation or policy function approximation where a "global" policy (one containing all-state policy information) [4].

The paper presents a unique combination of CFR and MCTS that can accelerate the time required to execute defensive measures against a cyberattack. A loss function or cost function is a function in mathematical optimization and decision theory that maps an event or values of one or more variables into a real number that intuitively reflects some "cost" associated with the event. An issue with optimization aims at reducing loss. An objective function is to be optimized by either a loss feature or its negative feature (in particular domains, a fee feature, a benefit feature, a utility function, a health function, etc).

Background

Related Work

In order to optimize an objective function (also known as a loss function), information-gathering concerns can be interpreted as sequential decision processes in which actions are chosen. Using myopic solvers (local minimum finder) that optimize the objective function over a limited time horizon usually overcomes the computational burden of decentralized coordination. Unfortunately, in general, the quality of solutions produced by myopic methods can be arbitrarily low[10]. However, submodularity analysis has recently shown that myopic techniques can achieve near-optimal efficiency, which has led to considerable interest in their application for data collection with multiple agents. Whereas theoretical guarantees are given by these greedy methods, they require a submodular objective function, which is not applicable in all cases. Furthermore, although these approaches also guarantee lower limits on optimality, by preparing over longer horizons (time intervals), the solution consistency can usually be increased. Similarly, decentralized task allocation approaches are also appropriate for simpler issues that only involve choosing one action per agent rather than sequences of actions. In general, the decentralized active knowledge gathering can be seen as a partially measurable Markov decision. In decentralized type, method (POMDP) (Dec-POMDP). Dec-POMDP formulations are usually solved through centralized, offline preparation over the joint multi-agent policy space (zero-sum game), and then these techniques are applied in a decentralized manner online [3]. In situations with broad sources of ambiguity, such as when the state of the environment is unknown in advance, these centralized, offline planning methods are impractical.

A study carried out by Keith & Ahner (2020) [12] on new application of optimal and approximate solution techniques to solve resource allocation problems with imperfect information in the cyber and air-defense domains. This model developed a two-player, zero-sum, extensive-form game to model attacker and defender roles in both physical and cyber space with aims of reformulating the problem to find a Nash equilibrium using an efficient, sequence-form linear program. Solving this linear program produces optimal defender strategies for the multi-domain security game. However, this model only addressed large problem instances with an application of the approximate counterfactual regret minimization algorithm as it reduces computation time by 95% while maintaining an optimality gap of less than 3%. This discounted counterfactual regret results in a further 36% reduction in computation time from the base algorithm helping to generate domain insights through a designed experiment to explore the parameter space of the problem and algorithm. We also address robust opponent exploitation by combining existing techniques to extend the counterfactual regret algorithm to include a discounted, constrained variant. A comparison of robust linear programming, data-biased response, and constrained counterfactual regret approaches clarifies trade-offs between exploitation and exploitability for each method leading to the understanding that linear programming approach is the most effective, producing an exploitation to exploitability ratio of 10.8 to 1.

On the other hand, Schmid et al., (2019) [13] on variance technique (VR-MCCFR) found out that the use of MCTS allows estimates to be bootstrapped from other estimates within the same episode, propagating the benefits of baselines along the sampled trajectory; the estimates remain unbiased even when bootstrapping from other estimates. This shows that given a perfect baseline, the variance of the value estimates can be reduced to zero. Experimental evaluation shows that MCTS brings an order of magnitude speedup, while the empirical variance decreases by three orders of magnitude. The decreased variance allows for the first time CFR to be used with sampling, increasing the speedup to two orders of magnitude.

Spaan et al. (2006) [8], on the other hand, presented a Dec-POMDP problem environment in which both preparation and execution are carried out in a decentralized manner; we approach our problem in a similar decentralized environment in such a way that computation is carried out online and on-board the agent Spaan et al. (2006) suggested a general Dec-POMDP solver where each agent solves a POMDP single-agent, shares its own plan details, then repeats or loops. Different agents play a different role in POMDP with attacker and agents of attack sharing same tree patterns.

Although the type of problem considered in Counterfactual Regret Minimization and Monte Carlo Tree Search for Cybersecurity Threats, it is not formulated in general as a Dec-POMDP, extended algorithms could be constructed using partially observable Monte Carlo planning for the Dec-POMDP situation.

MCTS has recently become famous for online planning. In several different ways, MCTS has been suggested (Browne et al., 2012) [9], but the upper-confidence bounds applied to trees (UCT) algorithm are by far the most common. Using a best-first policy which generalizes the UCB1 policy for multi agent bandit or MAB issues, the UCT algorithm performs an asymmetric expansion of a search tree. The theoretical guarantees for a polynomial bound on regret are given by this expansion policy and are therefore said to balance exploration and exploitation. Several variants of UCT have been suggested, such as leveraging the reward function's smoothness. A novel UCT version, D-UCT, is a key component of Dec-MCTS algorithm (whe is the proposed algorithm???) that accounts for an evolving distribution of rewards by using a new expansion strategy that generalizes the D-UCB policy for cyber threats detection. For partial-observability problems, such as the POMCP algorithms, MCTS algorithms have also been extended and Dec-MCTS may be extended in a similar algorithm.

In his work Neller & Lanctot (2013)[11] used Counterfactual Regret Minimization (CFR) is as worked example in solving Kuhn Poker and Rock-Paper-Scissors (RPS). RPS has three gestures: rock (closed fist), paper (an open facedown palm) and scissors (two extension fingers). The RPS illustrates how close linkage of how one addition forces in a game must have one additional probability of reaching information in a given set. The poker concept of randomization can be translated to probability of cyber system attack where distribution is not uniform. However, Neller &Lanctot (2013)[11] realized that such operation was not possible without the assumption that players only play in axed bounded policy error policy.

Systems and Metrics

An important topic in this paper is the Dec-POMDP or Decentralized Partially Observable Markov Decision Process which is an agent decision process that assumes that the system dynamics are determined by an Markov Decision Process(MDP), but the agent cannot directly observe the underlying state. Instead, it must maintain a probability distribution over the set of possible states, based on a set of observations and observation probabilities, and the underlying MDP. Dec-POMDP is modeled by the tuple $(N,\ S, \vec{A},\ T, \vec{R}, \vec{O}, Z, \gamma)$, where N is the number of agents. S is a set of states. $\vec{A} = A_1 \times A_2 \times A_i$ is the set of *joint action*, and A_i is a set of local action that agent i can take. $T(s'|s, \vec{a}) : S \times \vec{A} \times S \rightarrow [0,1]$ represents the state transition. The joint action $\vec{a} = (a_i, \vec{a}_{-i\_})$ results in a new state s' and a joint reward $\vec{r} = [r_1, \dots, r_N]$, where $\vec{a}\_i$ is the joint action of teammates of agent i. $\vec{R} = [R_1, \dots, R_N] : S \times \vec{A} \rightarrow \mathbb{R}^N$ is the *joint reward* function. $\vec{O} = O_1, \dots, O_N$ is the set of *joint observations* controlled by the observation function $Z: S \times \vec{A} \rightarrow \vec{O}$. $\gamma \in [0,1]$ is the discount factor. Dec-POMDP can model cybersecurity games.

Also, an important metric that measures the success of a decision making model is the regret. Standard regret is represented by the equation: $R_i^T = \frac{1}{T} \max_{\sigma_i^* \in \Sigma i} \sum_{t=1}^{T} \left(u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t) \right)$. Regret contains the maximum of the difference between utilities. Formulations that minimize the regret are most advantageous. An example of regret is counterfactual regret represented by the equation: $R_{i,imm}^T(I) = \frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(I) \left(u_i(\sigma^t|_{I \rightarrow a}, I) - u_i(\sigma^t, I) \right)$. The term $u_i(\sigma, I)$ represents counterfactual utility, where utility measures the value of an outcome. During a game either a high or low utility in a game can measure the success of a set of strategies depend on the perspective of the players of the game.

Monte Carlo Tree Search or MCTS is a family of algorithms that influence regret by maximizing reward. MCTS has four phases: selection, expansion, simulation and backpropagation. During the selection phase the algorithm processes the branch if the current branch is not a terminal leaf. The Upper Confidence Bound 1 with Trees or UCT is used to select the nodes.

The formula for UCT: $UCT = \frac{w_i}{n_i} + c \times \sqrt{\frac{\ln N_i}{n_i}}$. w_i stands for the number of wins for the node considered after the i-th move. n_i stands for the number of simulations for the node considered after the i-th move.

N_i stands for the total number of simulations after the i-th move run by the parent node of the one considered. c is the exploration parameter—theoretically equal to $\sqrt{2}$ in practice usually chosen empirically.

The second phase of MCTS is the expansion phase. During the expansion phase, a new child node is added to the tree that node which was optimally reached during the selection process. After the expansion phase is the simulation phase. During the simulation phase a strategy is executed until a result or predefined state is achieved. Starting from the position of the child node, the simulation makes random moves repeatedly until the game is won or lost. The last phaseof MCTS is backpropagation. During backpropagation, the function backpropagates from the new node to the root node. After determining the value of the newly added node, the remaining tree nodes must be updated. During the process, the number of simulations stored in each node is incremented. Also. If the simulation of the new node results in a win, then the number of wins is also incremented. Following backpropagation, the algorithm returns to the selection phase and loops through the four phases until a path of highest wins emerges and the iterations run out.

Main Findings and Numerical Results

To measure the performance of counterfactual regret minimization (CFR) in comparison to discounted counterfactual regret minimization (DCFR)'s variable of exploitability. Exploitability is a count of the number of chips required to beat an opponent. For example, suppose a random strategy is employed and an exploitability of 175 is computed. This means that a worst-case opponent can beat the random strategy for an average of 175 chips. The higher the exploitability value the worse the strategy and the lower the exploitability value the better. When the exploitability for CFR is computed in comparison to a random strategy, CFR obtains an exploitability of around 1. Furthermore, when the exploitability for DCFR is computed in comparison to CFR, DCFR obtains an exploitability between 1 and 0. With the lower exploitability values of DCFR, one may argue DCFR has a better strategy in comparison to CFR that helps in defining paths exploited by threats.

CFR	DCFR
1.05092216	0.00189
1.10765839	0.00059
1.03305769	0.00057
1.08745146	0.0003
1.26718855	0.0002

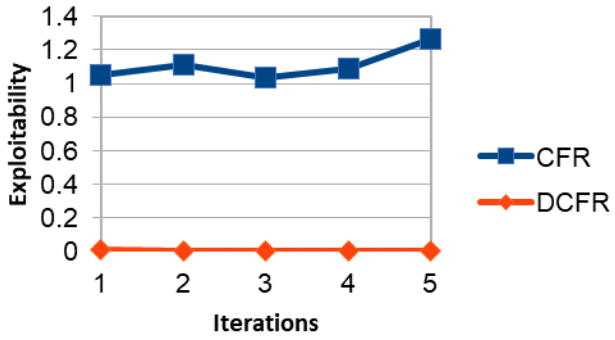

Figure 1: CFR and DCFR comparison of Exploitability

The MCTS path with the highest number of wins represents the best path. If a hacker attempts illicit access to a network, then there would be a deflection of the path. There are six sets of paths for the default versus alternative path.

The most common path pattern occurs when the default path and the alternative path share common nodes besides the root node where the root node is common to all paths. When the default path and the alternate path share two nodes including the root node there appears to be an either downward drift or upward drift towards the terminal node. For example,suppose both the default path and the alternative path share the root and node two, when the bifurcation occurs at node two the default path follows a path parallel to the alternative path.

More explicitly, after the bifurcation at node two, the default path from the node four to node sixteen takes the lower node from each branch until the terminal node is reached. Likewise, for the alternative path after the bifurcation at node two, there is a downward drift from node three to the terminal node fourteen.

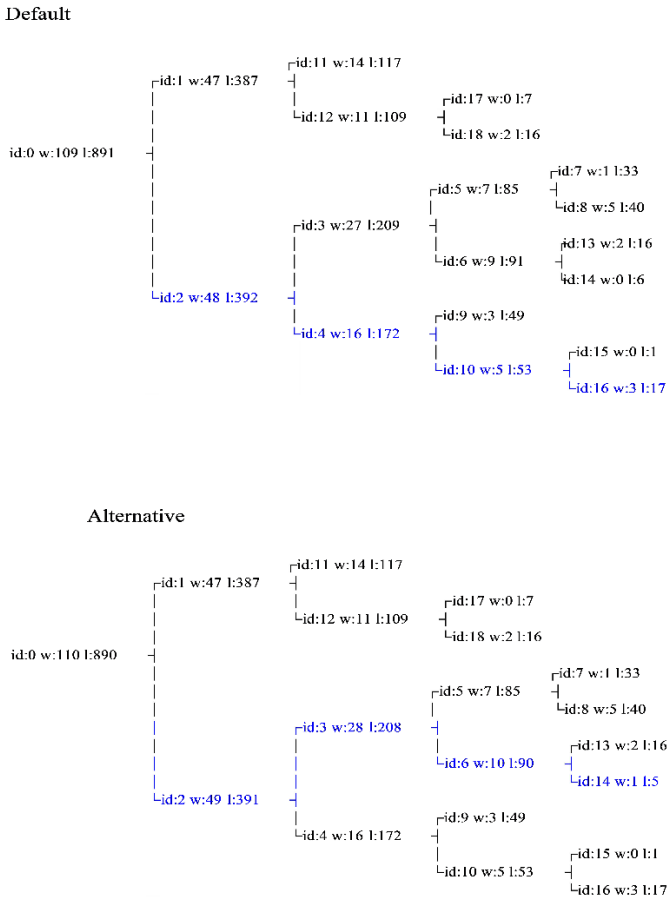

Figure 2: MCTS Pattern 1, Share Node with Bifurcation

Default Tree	Tree with alternative value
optimal path=[16,10,4,2]	optimal path=[14,6,3,2]
sum win=72	sum win=73

An equally interesting attacker behavior occurs when three nodes, including the root node, are shared between the default path and alternate path. When three nodes are shared there appears to be a pre-terminal deflection followed by a terminal downward drift in one path, while there is a pure downward drift in another path. For example, suppose a default path and alternative path share the nodes two and six in addition to the root, and there is a bifurcation at node six. The default path deflects upward to node thirteen and then downward to terminal node sixteen. While the alternative path maintains a downward trajectory and drift for node six to node fourteen and then to terminal node twenty.

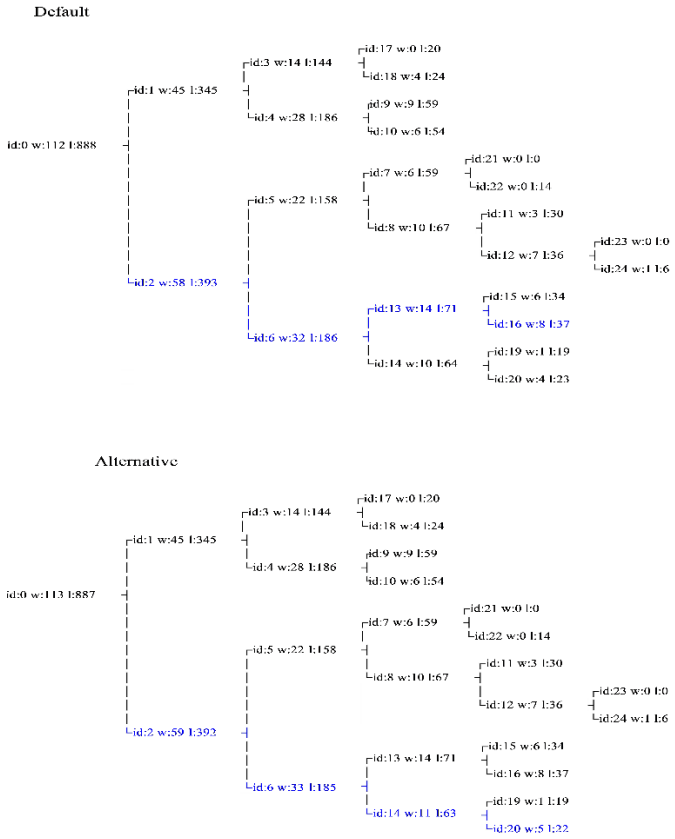

Figure 3: MCTS Pattern 2, Two Shared Nodes with Deflection

Default Tree	Tree with alternative value
optimal path=[16,13,6,2]	optimal path=[20,14,6,2]
sum win=112	sum win=114

For default paths and alternative paths that only share the root there appears to be divergent paths. As divergence occurs at the roots there is an upward drift of one path, while there is a downward drift for the other paths. With the divergence there occurs a partial path pre-terminal or terminal deflection. If you take for example a scenario where the default path runs from the root node and then to node two, then node four and finally node seven; that default path would have repeatedly selected the lower bound node with the exception of the deflection which occurred at the terminal node. On the other hand, the alternative path would run from the root node to node one, and then node five, then node nine, and then terminate at terminal node eleven. This alternative path has consistently chosen an upper node for each branch of the path, which is in deep contrast to the default path that mostly chose the lower nodes excluding the terminal.

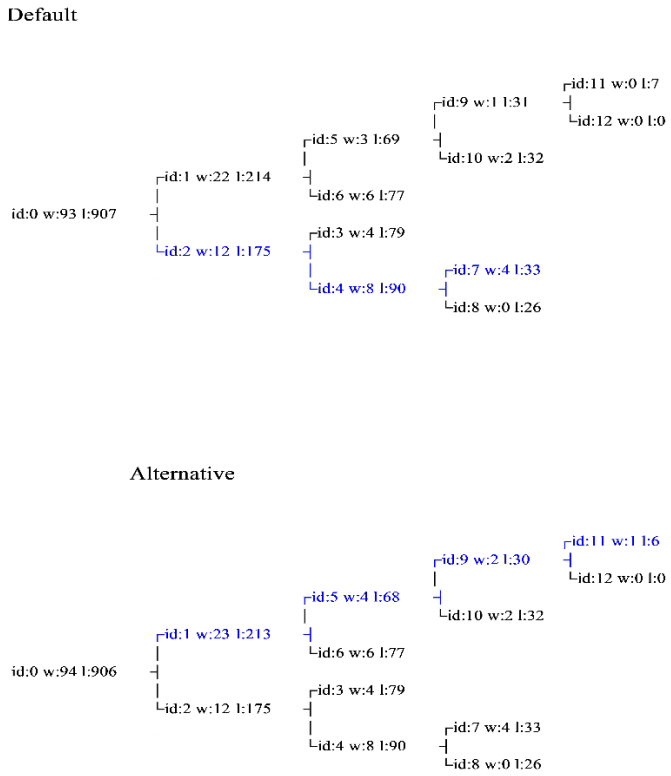

Figure 4: MCTS Pattern 3, Divergence

Default Tree	Tree with alternative value
optimal path=[7,4,2]	optimal path=[11,9,5,1]
sum win=24	sum win=24

In sharp contrast to the divergence of the default paths and alternate paths, there exist situations of convergence. When the default path progressively goes to higher nodes and the alternative path goes for successively lower nodes then convergence appears to manifest. For example, suppose the default path and alternate path bifurcate at the root node. The lower node representing the default path traverses an upward path and goes from node two to node three, while higher node synonymous with the alternate path, stems from the node one and charts a downward path to node twelve. However, in some cases after inner terminal node convergence occurs there can be episodes of emerging divergence. For example, after the default path converges upward from a downward node, there can be future divergence. The alternative path can diverge and can go upward to node thirteen.

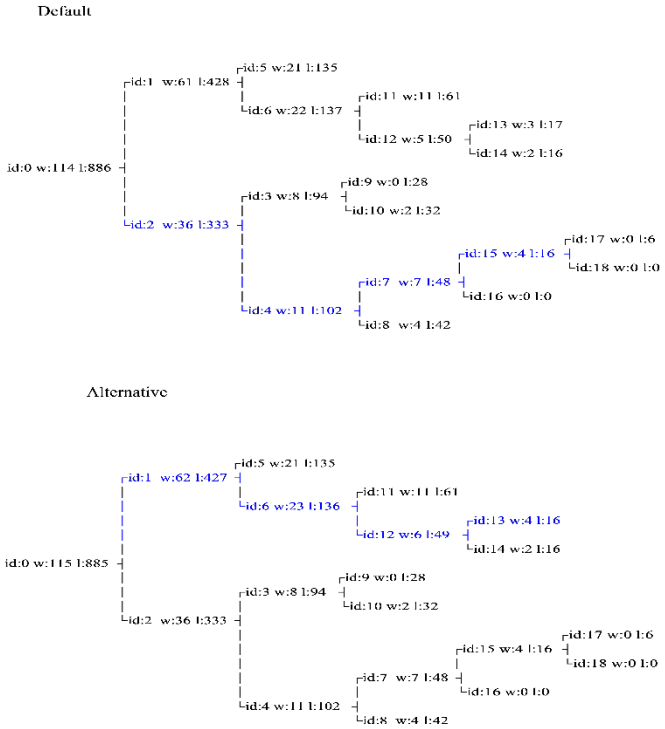

Figure 5: MCTS Pattern 4, Convergence

Default Tree	Tree with alternative value
optimal path=[15,7,4,2]	optimal path=[13,12,6,1]
sum win=58	sum win=58

Another major category of MCTS path patterns are the single shared node similars, which happen to share a root but then diverge. However, after divergence occurs, the paths of both the default path and the alternative path appear parallel. The parallelism is seen as both paths either drifting upwards by selecting higher order branches successively or drifting downwards by selecting lower order branches until a terminal node is reached. An example of parallel drift occurs when after bifurcation of the root node the default path moves up from root to node one, and then from one to seven, and finally to terminal node. While the alternate path starting at the root goes to node two, and then goes up to node three, followed by node five and finally terminating at node nine.

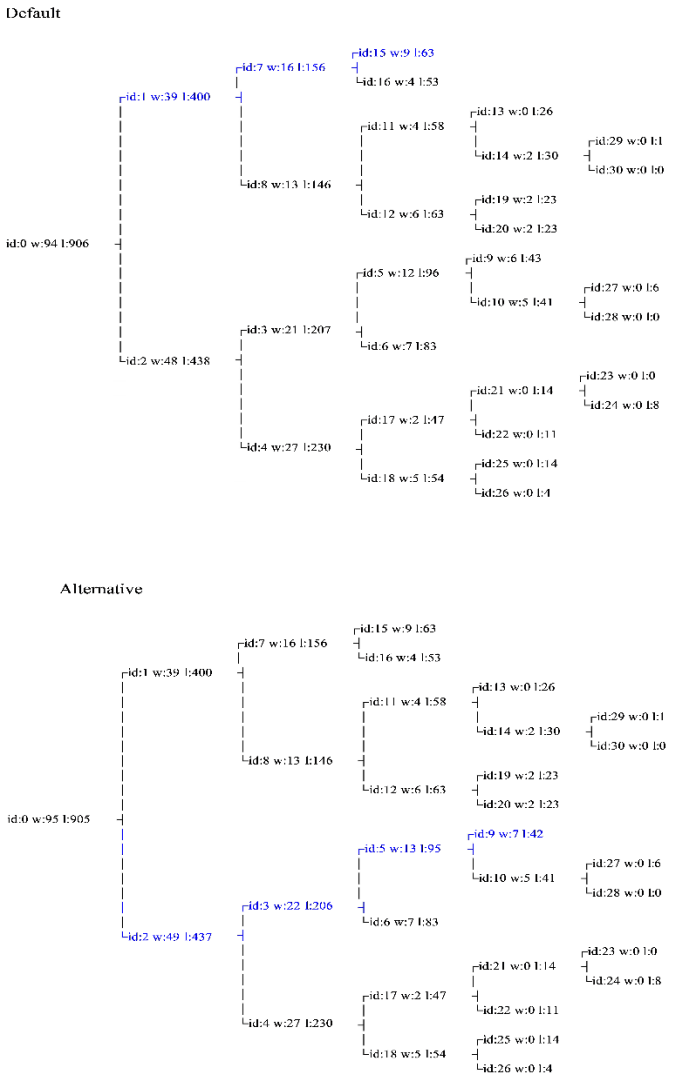

Figure 6: MCTS Pattern 5, Parallelism

Default Tree	Tree with alternative value
optimal path=[15,7,1]	optimal path=[9,5,3,2]
sum win=64	sum win=64

The last major class of MCTS paths patterns are the identical paths. When the default path and the alternate path share all the same nodes, from root node to terminal branch then the paths are said to be identical. This is a strict definition compared to the other MCTS path patterns because the path patterns are not dogmatically adhered to. The preceding path patterns served to provide a framework to understand the behavior of MCTS. As nothing in life is perfect the decision models of MCTS behavior fail to be perfect. These decision models however do provide a convention and vocabulary that help understand, predict and control MCTS; and therefore, create defensive measures.

New Discoveries

The combination of MCTS and CFR presented in this paper provide superior protection in comparison to the MCTS alone or CFR alone. By using CFR with MCTS there is earlierdetection of compromised resources. The detection time of the threat has the potential to be cut from down from months into weeks and therefore presents an essential alternative.

Conclusion

In summary, after repeatedly running the MCTS code and outputting default paths against alternate paths, the preceding patterns occurred. These pattern models are not perfect predictors of MCTS behavior and there exist many variants of the models described. However, combining the pattern models with the reasonable variants can cover most scenarios or situations where resources have been compromised.

With more time and more resources, more rigorous insight into MCTS can be seen. However, the argument can be made that after modifying the code of MCTS there is a modification of the optimal path of MCTS.

References

G. Best, O. Cliff and T. Patten, "Dec-MCTS: Decentralized planning for multi-robot active perception", The International Journal of Robotics Research, vol. 38, no. 2-3, 2018. Available: https://doi.org/10.1177/0278364918755924 [Accessed 25 January 2021].

T. Vodopivec, S. Samothrakis and B. Ster, "On Monte Carlo Tree Search and Reinforcement Learning", Journal of Artificial Intelligence Research, vol. 60, pp. 881-936, 2017. Available: 10.1613/jair.5507.

M. Zinkevich, M. Johanson, M. Bowling and C. Piccione, "Regret Minimization in Games with Incomplete Information", 2021. [Accessed 25 January 2021].

H. Mao et al., "Neighborhood Cognition Consistent Multi-Agent Reinforcement Learning", Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 05, pp. 7219-7226, 2020. Available: 10.1609/aaai.v34i05.6212.

L. Lu, W. Zhang, X. Gu, X. Ji and J. Chen, "HMCTS-OP: Hierarchical MCTS Based Online Planning in the Asymmetric Adversarial Environment", *Symmetry*, vol. 12, no. 5, p. 719, 2020. Available: 10.3390/sym12050719.

N. Vien and M. Toussaint, "Hierarchical Monte-Carlo Planning", in *The Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

[7]N. Zerbel, "Multiagent Monte Carlo Tree Search with Difference Evaluations and Evolved Rollout Policy", 2018. Available: 10.13140/RG.2.2.12916.81288 [Accessed 25 January 2021].

Spaan MTJ, Gordon GJ and Vlassis N, Decentralized planning under uncertainty for teams of communicating agents. In: *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems*, 2006.

Browne C, Powley E, Whitehouse D, et al A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4(1): 1–43, 2012.

Best, G., Cliff, O., Patten, T., Mettu, R., & Fitch, R. (2018). Dec-MCTS: Decentralized planning for multi-robot active perception. The International Journal Of Robotics Research, 38(2-3), 316-337. doi: 10.1177/0278364918755924

Neller, T., & Lanctot, M. (2013). An Introduction to Counterfactual Regret Minimization, Gettysburg College, Department of Computer Science, Campus Box 402, Gettysburg, PA17325-1486

A. Keith and D. Ahner, "Counterfactual regret minimization for integrated cyber and air defense resource allocation", European Journal of Operational Research, 2020. Available: 10.1016/j.ejor.2020.10.015.

M. Schmid, N. Burch, M. Lanctot, M. Moravcik, R. Kadlec and M. Bowling, "Variance Reduction in Monte Carlo Counterfactual Regret Minimization (VR-MCCFR) for Extensive Form Games Using Baselines", Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 2157-2164, 2019. Available: 10.1609/aaai.v33i01.33012157.