Stackelberg Equilibrium Approximation in General-Sum Extensive-Form Games with Double-Oracle Sampling Method

Extended Abstract

Jan Karwowski
Warsaw University of Technology, Faculty of Mathematics
and Information Science
Warsaw, Poland
jan.karwowski@mini.pw.edu.pl

Jacek Mańdziuk
Warsaw University of Technology, Faculty of Mathematics
Information Science
Warsaw, Poland
mandziuk@mini.pw.edu.pl

ABSTRACT

The paper presents a new method for approximating Strong Stackelberg Equilibrium in general-sum sequential games with imperfect information and perfect recall. The proposed approach is generic, i.e. does not rely on any specific properties of a particular game model. The method is based on iterative interleaving of the two following phases: (1) guided Monte Carlo Tree Search sampling of the Follower's strategy space and (2) building the Leader's behavior strategy tree for which the sampled Follower's strategy is an optimal response. The above solution scheme is evaluated on interception games played on graphs with respect to expected Leader's utility and time requirements. A comparison with two state-of-the-art exact methods for this genre of games shows that in vast majority of test cases our simulation-based approach leads to optimal Leader's strategies, while excelling both exact methods in terms of time scalability and much lower memory usage.

KEYWORDS

Game Theory; Noncooperative games: computation; UCT; Guided Monte Carlo Tree Search

ACM Reference Format:
Jan Karwowski and Jacek Mańdziuk. 2019. Stackelberg Equilibrium Approximation in General-Sum Extensive-Form Games with Double-Oracle Sampling Method. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

Majority of contemporary Stackelberg Game (SG) research is focused on developing effective methods for specific game definitions, e.g. [2, 5, 10, 16, 20] and there are just a few works related to finding SE in the case of general SG models. Possible approaches include: *column and constraint generation* [8, 20], *marginal and compact strategies* [13, 16], *game abstraction* [1, 20] or *memetic algorithm* [12], however, none of them can be easily applied to a broad class of sequential multi-act general-sum games with imperfect information. An efficient exact approach to generic sequential general-sum SGs was proposed in [3] (referred to as *BC2015*) where the authors considered a sequence-form game representation. Another powerful general approach [7] (referred to as *Cermak2016*) starts off with finding Correlated Equilibrium using MILP and then restricts it iteratively until the SE strategy profile is obtained. These two state-of-the-art generic methods are reference approaches for an approximate method proposed in this paper.

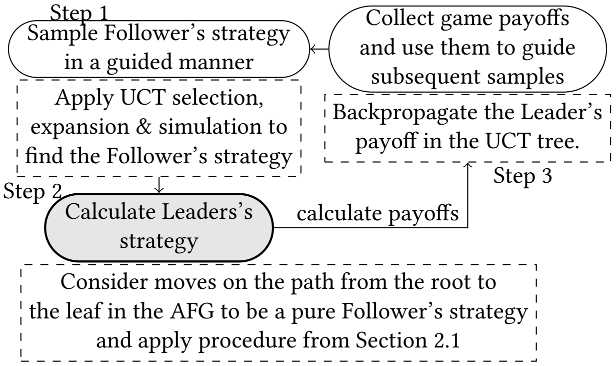
Step 1
Sample Follower's strategy in a guided manner
Collect game payoffs and use them to guide subsequent samples
Apply UCT selection, expansion & simulation to find the Follower's strategy
Backpropagate the Leader's payoff in the UCT tree.
Step 3
Step 2
Calculate Leaders's strategy
calculate payoffs
Consider moves on the path from the root to the leaf in the AFG to be a pure Follower's strategy and apply procedure from Section 2.1

Figure 1: An outline of the O2UCT method.

Contribution. The paper introduces a method for approximating SE in a broad and general genre of sequential general-sum imperfect-information games, inspired by a double-oracle approach [4, 9]. Despite being rooted in the double-oracle framework, the method presents an entirely different operational principle as it relies on Upper Confidence bound applied to Trees (UCT) [14] - a variant of Monte Carlo Tree Search (MCTS) [6] sampling of the Follower's strategy alternated with an adjustment of the Leader's behavior strategy represented in the form of a tree.

2 DOUBLE-ORACLE SAMPLING METHOD (O2UCT) FOR SE APPROXIMATION

The proposed approach, called O2UCT (double-oracle UCT sampling), aims at approximating Leader's equilibrium strategy in sequential general-sum games with perfect recall and imperfect information. An overview of the method is depicted in Figure 1. A distinctive feature of O2UCT is the lack of exhaustive search of the Follower's strategy space, which is replaced by iterative guided space sampling. In principle, any sampling method capable of transferring knowledge about the sampled space to subsequent iterations can be used. In this paper the UCT method, which already proven successful in a wide variety of domains [15, 17–19], is applied.

In short, each UCT iteration (playout) is composed of 4 main phases: *selection*, *expansion*, *simulation*, and *backpropagation* [6, 14].

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. © 2019 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

```
Algorithm 1: Node adjustment with momentum
Data: $prob \in [0,1]^M$ – a vector of probabilities, $mom \in \mathbb{R}^M$ –
      a momentum vector, $w \in \mathbb{R}$ – a normalization factor,
      $as \in \mathbb{R}^M$ – an assessments vector.
$mom \leftarrow mom + as$;
$w \leftarrow w + L_1(as)$;
$prob \leftarrow \max\{prob + mom/w, 0\}$// independent max at
    each position
$prob \leftarrow normalizeOrEqualprob$// Normalize vector
    values so their sum :=1 or assign equal prob.
    at each position if current sum is 0
```

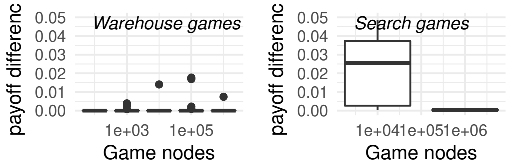

Figure 2: A difference in Leader's payoffs between the SSE strategy and the O2UCT strategy w.r.t. the number of nodes.

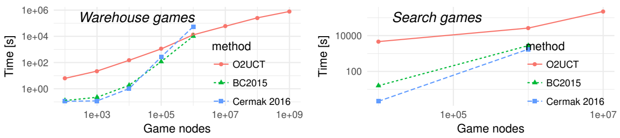

Figure 3: Computation times comparison.

In our method, selection, expansion and simulation correspond to Step 1 of O2UCT iteration (guided sampling in Figure 1), and backpropagation is implemented in Step 3 (collection of payoffs). Step 2 refers to obtaining the Leader's strategy, **for which the just-sampled Follower's strategy is the optimal response**. The expected Leader's payoff is calculated using a method presented in Section 2.1.

2.1 A method of finding the Leader's strategy

The algorithm for finding Leader's strategy is inspired by a double-oracle approach [4, 9] and consists in alternating the following two phases: (1) an improvement of the Leader's strategy against a fixed Follower and (2) finding the optimal Follower's response against the current Leader's strategy based on the Follower's oracle. For a sampled Follower's strategy (Step 1 in Fig. 1) a corresponding Leader's strategy (Step 2 in Fig. 1) must satisfy two conditions:

(*) the optimal Follower's response to that strategy is the same as the sampled Follower's strategy;

(**) among all Leader's strategies that satisfy (*) it is the one that optimizes the Leader's payoff.

Any Leader's strategy satisfying (*) will be called a *feasible strategy*.

Let's denote the sampled Follower's strategy by δ_F^r (r stands for requested Follower's strategy). The method of finding a strategy that approximates (*)-(**) consists of the following steps:

(1) Initialize Leader's strategy.
(2) Seek the Follower's strategy yielding better Follower's payoff against the current Leader's strategy. If exists, call it δ_F^b.
(3) If δ_F^b was found, then perform strategy *feasibility pass* (see below) and go to (2), otherwise continue.
(4) Perform the Leader's strategy adjustment that increases the Leader's payoff (*positive pass* - see below) and go to (2).
(5) Return the best Leader's strategy among all *feasible strategies* found (in step (3)).

Leader's payoff improvement (4) is repeated until either iteration-to-iteration Leader's payoff increase is smaller than a pre-defined threshold or the limit for the total number of iterations is reached.

All adjustments to the Leader's strategy are performed in each node of a continuously evolving tree-based representation, according to Algorithm 1, starting from the bottom of the tree, in one of the two following procedures: *feasibility pass* and *positive pass*. The first one is executed when there exists δ_F^b that yields higher Follower's payoff than δ_F^r and changes Leader's move probabilities to increase the value of $\delta_F^r - \delta_F^b$. The latter one is run to improve the Leader's payoff in the case of feasible (Leader's) strategy.

3 EXPERIMENTAL EVALUATION

Efficiency and scalability of O2UCT was compared with *BC2015* and *Cermak2016* methods, on two game sets: Warehouse Games [11] and Search Games [3], on Intel Xeon Silver 4116 @ 2.10GHz with 256GB RAM and time limit of 200 hours. The first game family was tested with the number of rounds $T = 3, 4, 5, 6, 7, 8$, the second one for $T = 4, 5, 6$. For *Cermak2016* a variant called AI-MILP was used [7]. For each game instance between 5 and 15 O2UCT tests were run. The baseline methods were run once as they have deterministic nature. Performance of O2UCT is analyzed in two dimensions: an expected Leader's payoff (Figure 2) and computation time (Figure 3). The outcomes are grouped by the number of nodes of an extensive-form game representation. While exact measurements of memory usage were not performed (because of using a garbage collector) we noted that O2UCT was able to compute results for 10^9 game nodes using 8GB of memory while solver based methods started running out of (256GB) memory for games with 10^7 nodes.

4 CONCLUSIONS

O2UCT provides high-quality solutions – optimal in vast majority of test cases, while scaling visibly better than exact state-of-the-art MILP-based methods. The method is capable of solving longer / more complex game instances due to lower memory requirements, which stem from two factors: application of a double oracle approach (which does not require storing in memory all possible strategy profiles), and dynamic UCT-based expansion of the Leader's strategy tree. Furthermore, the UCT-based sampling is an anytime procedure which can be stopped in any moment, though still returning a high quality solution (the best one found so far). Finally, O2UCT is a generic method applicable to any sequential games.

ACKNOWLEDGMENT

This work was supported by the National Science Centre, grant number 2017/25/B/ST6/02061.

REFERENCES

[1] Anjon Basak, Fei Fang, Thanh Hong Nguyen, and Christopher Kiekintveld. 2016. Abstraction Methods for Solving Graph-Based Security Games. In *Autonomous Agents and Multiagent Systems*. Springer International Publishing, Singapore, 13–33.

[2] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. 2012. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial Intelligence* 184–185 (Jun 2012), 78–123. https://doi.org/10.1016/j.artint.2012.03.003

[3] Branislav Bosansky and Jiri Cermak. 2015. Sequence-Form Algorithm for Computing Stackelberg Equilibria in Extensive-Form Games. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press, Austin, 805–811. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9610

[4] Branislav Bosansky, Christopher Kiekintveld, Viliam Lisy, and Michal Pechoucek. 2014. An Exact Double-Oracle Algorithm for Zero-Sum Extensive-Form Games with Imperfect Information. *J. Artif. Intell. Res.* 51 (2014), 829–866. https://doi.org/10.1613/jair.4477

[5] Tomas Brazdil, Antonin Kucera, and Vojtech Rehak. 2018. Solving Patrolling Problems in the Internet Environment. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, Stockholm, Sweden, 121–127. https://doi.org/10.24963/ijcai.2018/17

[6] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *Computational Intelligence and AI in Games, IEEE Transactions on* 4, 1 (2012), 1–43.

[7] Jiri Cermak, Branislav Bosansky, Karel Durkota, Viliam Lisy, and Christopher Kiekintveld. 2016. Using Correlated Strategies for Computing Stackelberg Equilibria in Extensive-Form Games. In *30th AAAI Conference on Artificial Intelligence*. AAAI Press, Phoenix, 439–445.

[8] Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordónez, and Milind Tambe. 2010. Security Games with Arbitrary Schedules: A Branch and Price Approach. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press, Atlanta, USA, 792–797.

[9] Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. 2011. A double oracle algorithm for zero-sum security games on graphs. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Taipei, Taiwan, 327–334.

[10] Matthew Paul Johnson, Fei Fang, and Milind Tambe. 2012. Patrol Strategies to Maximize Pristine Forest Area. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press, Toronto, Canada, 295–301. http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4745

[11] Jan Karwowski and Jacek Mańdziuk. 2019. A Monte Carlo Tree Search approach to finding efficient patrolling schemes on graphs. *European Journal of Operational Research* (Feb 2019). https://doi.org/10.1016/j.ejor.2019.02.017

[12] Jan Karwowski, Jacek Mandziuk, Adam Zychowski, Filip Grajek, and Bo An. 2019. A Memetic Approach for Sequential Security Games on a Plane with Moving Targets. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*. AAAI Press, Honolulu, USA.

[13] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. 2009. Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, Budapest, Hungary, 689–696.

[14] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based Monte-Carlo planning. In *Machine Learning: ECML 2006*. Springer, Cham, Switzerland, 282–293.

[15] Ashish Sabharwal, Horst Samulowitz, and Chandra Reddy. 2012. Guiding combinatorial optimization with UCT. In *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems*. Springer, Berlin, Germany, 356–361.

[16] Aaron Schlenker, Matthew Brown, Arunesh Sinha, Milind Tambe, and Ruta Mehta. 2016. Get Me to My GATE on Time: Efficiently Solving General-Sum Bayesian Threat Screening Games.. In *ECAI*. IOS Press, The Hague, The Netherlands, 1476–1484.

[17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (October 2017), 354–359. https://doi.org/10.1038/nature24270

[18] M. Świechowski, J. Mańdziuk, and Y-S. Ong. 2016. Specialization of a UCT-based General Game Playing Program to Single-Player Games. *Computational Intelligence and AI in Games, IEEE Transactions on* 8, 3 (2016), 218–228.

[19] Karol Walędzik and Jacek Mańdziuk. 2018. Applying hybrid Monte Carlo Tree Search methods to Risk-Aware Project Scheduling Problem. *Information Sciences* 460–461 (2018), 450–468. https://doi.org/10.1016/j.ins.2017.08.049

[20] Xinrun Wang, Bo An, Martin Strobel, and Fookwai Kong. 2018. Catching Captain Jack: Efficient time and space dependent patrols to combat oil-siphoning in international waters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, New Orleans, USA, 208–215. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16312