ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1746

Mixed Strategy Extraction from UCT Tree in Security Games

Jan Karwowski[1] and **Jacek Mańdziuk**[1 2]

Abstract. In this paper a simulation-based approach to finding optimal defender strategy in multi-act Security Games (SG) played on a graph is proposed. The method employs the Upper Confidence Bounds applied to Trees (UCT) algorithm which relies on massive simulations of possible game scenarios. Three different variants of the algorithm are presented and compared with each other as well as against the Mixed Integer Linear Program (MILP) exact solution in terms of computational efficiency and memory requirements. Experimental evaluation shows that the method has a few times lower memory demands and is faster than MILP approach in majority of test cases while preserving quality of the resulting mixed strategies.

1 INTRODUCTION

This paper describes simulation based approach to SG which can be used to find reasonably good strategies in SG with many rounds. Contrary to currently used methods which require a game to be represented in normal form or extensive form used in game theory, our method uses games represented by a set of game rules defining states, moves and results. This way we avoid memory demanding explicit payoff representations (typically used in SG) which assign a particular payoff for each possible move sequence and quickly become intractable as the number of possible move sequences grows exponentially with the number of rounds.

1.1 Security Games

SG is a field of science which applies mathematical models to patrolling schemes and other security operations in order to find optimal strategies for security forces against adversaries [3, 10, 7]. Possible applications include homeland security, fighting crime, or securing industrial objects. Usually SG are played by two sides: the defender (representing security forces) and the attacker (representing terrorists, criminals, etc.). The game is of imperfect information and players make their moves simultaneously. Instead of searching for the best move in a single game, the goal is to find a strategy (probability distribution of moves to play) for the defender that will maximize the expected reward. Almost all SG use Stackelberg Game [6] to model the game and are often referred as Stackelberg Security Games (SSG). Players in Stackelberg Game are asymmetric. One of them is called a Leader (the defender in SG) and the other one is a Follower (the attacker in SG). Asymmetry is introduced by the fact that the Follower knows the Leader's strategy before committing to their strategy. Such asymmetry models real-life cases in which the attacking side can observe the defending side sufficiently long to learn their strategy (probability distribution of actions).

Strong Stackelberg Equilibrium [6], which is the optimal solution for a defender can be expressed as a solution to bi-level optimization problem. The state-of-the-art solutions for SG transform this problem to a form suitable for popular Mathematical Programming solvers and use them to compute the strategy.

1.2 Upper Confidence Bounds applied to Trees

UCT method [5] is a variant of Monte Carlo Tree Search (MCTS) method used for searching for a single best move in perfect-information game tree [9, 8]. UCT uses large number of game simulations to estimate reward value for each move. Internally UCT keeps a tree consisting of game states (nodes) and actions or moves (edges). Each edge is labeled with two values: $Q(s,a)$ – current estimate of action a played in state s, and $N(s,a)$ – number of times the action was hitherto simulated. During simulations the method chooses actions based on the following formula:

$$a^* = \operatorname*{arg\,max}_{a\in A} \left\{ Q(s,a) + C\sqrt{(\ln N(s))/N(s,a)} \right\}, \qquad (1)$$

where $N(s) = \sum_a N(s,a)$ and C is the method's parameter. Formula (1), often called UCB1 [2], maintains a balance between exploitation of currently good estimated moves and exploration of rarely visited ones. After the simulation phase a move with the best Q estimation is chosen to be played.

2 Mixed-UCT METHOD

While UCT can be used for finding the best *single* move in perfect information game, the Mixed-UCT method, proposed in this paper,

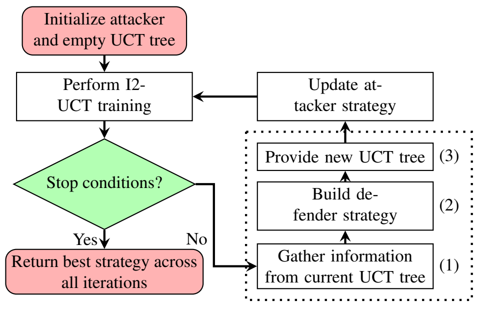
Initialize attacker and empty UCT tree
Perform I2-UCT training
Update attacker strategy
Provide new UCT tree
(3)
Build defender strategy
(2)
Stop conditions?
Yes
No
Return best strategy across all iterations
Gather information from current UCT tree
(1)

Figure 1. Outline of proposed Mixed-UCT method. Dotted line encompasses a generic procedure for the defender's strategy definition, which is the heart of the method.

[1] Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland, email {jan.karwowski, j.mandziuk}@mini.pw.edu.pl
[2] School of Computer Science and Engineering, Nanyang Technological University, Singapore

is a UCT-based approach to finding the best mixed strategy for the defender in SSG. It relies on a UCT modification, referred to as I2-UCT, applicable to the task of finding the *best move* in imperfect-information games [4]. Basically I2-UCT extends vanilla UCT by adding a method of sampling (estimating) the current node (position) in imperfect information games before the actual UCT simulations take place (see [4] for details).

The proposed Mixed-UCT method iteratively applies I2-UCT procedure in order to find the best *mixed strategy* in SSG. The outline of the method is presented in Figure 1. For the sake of space limits we will solely concentrate on the most interesting part of the method, i.e. transformation of the data gathered by I2-UCT (intended to find a single best move) into coherent mixed strategy. Three variants of such a transformation are proposed. Each of them consists of three procedures denoted by (1), (2) and (3) in the figure.

Single tree variant uses UCT statistics gathered in a single (common) tree during training against many attacker strategies. **(1)** simply stores the UCT tree obtained from the recently completed training. **(2)** works as follows: in the stored UCT tree each path from the initial state (tree root node) to a terminal state (a leaf node) is assigned a weight being a product of visit counters of all edges (moves) on this path. Each such path represents a sequence of moves in the game – i.e. a pure strategy. The resulting mixed strategy is a probability distribution of these pure strategies, where probability of each sequence (pure strategy) is proportional to its weight – the weights are normalized to sum up to 1. **(3)** provides the tree stored by (1).

Best path variant collects best move sequences, one per each attacker it played against. **(1)** finds the best path from the root to a leaf in the trained tree according to Q estimates and stores a move sequence (pure strategy) extracted from this path. **(2)** returns probability distribution of these collected move sequences with probabilities proportional to number of their occurrences in these stored paths. **(3)** provides an empty tree for each I2-UCT training.

In **Tree slice** variant **(1)** prunes a tree by removing all paths from a given node that have Q estimate lower than 50% of the best Q value assigned to this node. Such pruned tree is stored and called a *tree slice*. **(2)** is a two-step procedure. First all stored *tree slices* are joined into one tree by summing visit counters in respective vertices, then a procedure from **Single tree** variant is applied to this joint tree. **(3)** provides an empty tree for each subsequent training.

	Single tree		Tree slice		Best seq.		Uniform	Optimal	
game	*t* [s]	*sc*	*t* [s]	*sc*	*t* [s]	*sc*	*R*	*R*	*t* [s]
1	2382	1	1559	0.96	1344	0.99	-10.46	0.54	28327
2	2614	1	2117	0.99	1587	1	-7.21	0.08	110
3	3188	1	7643	0.99	2486	0.97	-13.97	-4.27	17394
4	2115	1	3191	1	1379	1	-13.97	-4.5	18734
5	2325	1	3359	1	2484	1	-0.87	2.58	283
6	2058	1	3183	1	2151	1	-9.29	-1.06	18579
7	2241	1	4062	1	2897	1	-4.61	0.9	4695
8	1973	1	2364	1	1579	1	-13.91	-4.87	5823
9	2058	1	2007	1	1261	1	-13.84	-6	5915
10	1393	1	1448	0.99	1585	0.99	-0.81	0.79	5519
11	2579	1	2416	1	2093	1	-9.23	-2.85	5928
12	1713	1	2686	1	1775	1	-4.55	0.17	5149

Table 1. Defender's scores and computation times (t) of presented variants of Mixed-UCT in 12 test games (each averaged over 10 trials) compared with optimal solver-based strategy and the result of uniformly playing defender. *Score (sc)* represents the relative payoff assessment, which is placed on a scale from 0 (Uniform player) to 1 (Optimal player).

3 EXPERIMENTAL RESULTS

Each of the three variants of the method was tested against a benchmark set of 12 patrolling games defined on a graph. In each round both players were moving their units to adjacent vertices or let them remain in the current node. The goal of the attacker was to reach particular target node, while the defender's task was to catch the attacker (on the way or in the target) by being in the same vertex as them. Each game was played on a different graph with various reward and penalty distributions in the vertices.

Table 1 presents the average results of 10 game repetitions in each of the 12 test games, compared to the state-of-the-art method (MILP solutions calculated with SCIP [1]) and to a simple attacker that uses uniform strategy. Memory usage was comparable across all test cases and equal to about 5GB in MILP and about 1.2 in Mixed-UCT.

The results show that quality of Mixed-UCT solutions is close to optimal in all variants. Computational times of Mixed-UCT are shorter than those of the solver and vary between variants. **Tree slice** tends to be the slowest and **Best path** the fastest, in most of the cases.

4 CONCLUSIONS

This work introduces Mixed-UCT method and presents its initial experimental evaluation in SG domain. The results indicate general suitability of UCT-based methods to finding optimal strategies in SG. The proposed method, due to more compact (rule-based) game representation requires significantly less memory than MILP, which is currently the state-of-the-art approach. The proposed method is also faster than MILP solver. The results show that **Tree slice** version is inferior to the two other variants both in time and results quality, but at the same time, are inconclusive about superiority among the two other variants. Promising initial results lay foundations for further development of the method, in particular its evaluation on larger game graphs and games with multiple units on each playing side.

REFERENCES

[1] Tobias Achterberg, 'SCIP: Solving constraint integer programs', *Mathematical Programming Computation*, **1**(1), 1–41, (July 2009).

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer, 'Finite-time analysis of the multiarmed bandit problem', *Machine learning*, **47**(2-3), 235–256, (2002).

[3] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordóñez, 'Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service', *Interfaces*, **40**(4), 267–290, (2010).

[4] Jan Karwowski and Jacek Mańdziuk, 'A new approach to security games', in *International Conference on Artificial Intelligence and Soft Computing*, volume 9120 of *LNAI*, 402–411, Springer-Verlag, (2015).

[5] Levente Kocsis and Csaba Szepesvári, 'Bandit based monte-carlo planning', in *Machine Learning: ECML 2006*, 282–293, Springer, (2006).

[6] George Leitmann, 'On generalized stackelberg strategies', *Journal of Optimization Theory and Applications*, **26**(4), 637–643, (1978).

[7] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer, 'PROTECT: A deployed game theoretic system to protect the ports of the United States', in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 13–20, (2012).

[8] Maciej Świechowski and Jacek Mańdziuk, 'Self-adaptation of playing strategies in General Game Playing', *Computational Intelligence and AI in Games, IEEE Transactions on*, **6**(4), 367–381, (2014).

[9] Karol Waledzik and Jacek Mańdziuk, 'An automatically-generated evaluation function in General Game Playing', *Computational Intelligence and AI in Games, IEEE Transactions on*, **6**(3), 258–270, (2014).

[10] Yue Yin, Bo An, and Manish Jain, 'Game-theoretic resource allocation for protecting large public events', in *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI14)*, pp. 826–834, (2014).