A Monte Carlo Tree Search approach to finding efficient patrolling schemes on graphs

Jan Karwowski*, Jacek Mańdziuk

Faculty of Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Poland

Abstract

In this paper we propose an evader-defender type of game for modeling multi-step patrolling scenarios on a graph. The game utilizes a specifically designed graph-based setting which captures spatial arrangements of the protected area, for instance industrial premises or warehouses, wherein certain valuable assets are stored. The game is played by two sides: *the evader* who attempts to steal or destroy the assets and *the defender* whose aim is to intercept the evader and prevent him/her from accomplishing his/her goal.

Real-life specificity of the proposed game assumes *information asymmetry* between the two sides as the evader can usually observe defender's patrolling schedules prior to making decision of an attack. For this reason, we employ the Stackelberg Game principles to model our game and consequently focus on approximation of Stackelberg Equilibrium during the solution process. To this end we propose a novel approach, called Mixed-UCT, which relies on Upper Confidence Bound applied to Trees algorithm – a variant of Monte Carlo Tree Search.

The efficacy of the proposed solution method is experimentally evaluated on randomly generated games played in warehouse-like, industrial environment. The results show that Mixed-UCT is efficient and scales very well for *multi-step* games with reasonable number of steps, leading to optimal or close-to-optimal strategies.

Keywords: Game theory, Stackelberg Games, MCTS, Security Games

1. Introduction

In many real-life situations, effective planning of patrolling schedules is critical for ensuring adequate level of security for assets, locations or people (Haveman et al., 2005; Tambe, 2011). While such a task can be (and

*Corresponding author

Email addresses: jan.karwowski@mini.pw.edu.pl (Jan Karwowski), J.Mandziuk@mini.pw.edu.pl (Jacek Mańdziuk)

often is) performed by a human, using an algorithmic approach offers several advantages:

- it may potentially lead to discovering of highly effective, though counter-intuitive for humans, patrolling schemes (Neuringer, 1986).
- planned-by-humans patrolling schedules tend to be schematic, while algorithmically developed schemes, especially when backed-up by game-theoretic models, can lead to higher patrolling effectiveness (Fang et al., 2013; Jain et al., 2010b; Wang et al., 2018),
- human assessment of a security situation may not be fully rational and is often biased by some kind of subjectivity (Rubinstein, 1998).

In this paper, a patrolling-based approach to the problem of protection of certain areas, like warehouses or factory premises, modeled on graphs, is considered. On a general note, the considered environment admits a complex structure and includes valuable assets (e.g. commodities or machines), stored in certain places, which require special care from the patrolling security troops.

In order to capture the properties of such an environment, an undirected graph-based representation of locations is used. Graph vertices correspond to particular locations in the area. Edges denote the possibility of moving between the respective adjacent locations. Some locations, as mentioned above, contain certain assets that need to be protected from unauthorized access.

The solution relies on game-theoretic modeling of optimal responses. Patrolling scenarios are represented in the form of a sequential game played by two sides: *the evader* (attacker) who tries to reach the assets located in graph vertices, and *the defender*, possessing one or more patrolling units, whose goal is to intercept the evader. As patrolling is a repetitive activity, the defender's goal is to maximize the average expected utility value across infinitely many games, *not an outcome of a single sequential playout.*

A formal description of a game is presented in the next section. In short, it follows the Stackelberg Game (SG) model, which assumes *information asymmetry* between the players in the form that the evader knows the defender's mixed strategy (probability distribution of actions) before committing to a particular attack-related decision. This property, and consequently selection of SG as a baseline model, is motivated by the fact that in our settings an evader, during attack preparation, can surveil patrolling schedules of the defender and infer their probability distribution. The SG model has proven suitable in many similar applications, e.g. (Korzhyk et al., 2011; Johnson et al., 2012; Fang et al., 2013; Gan et al., 2015).

The proposed solution method (called *Mixed-UCT*) relies on approximation of the Stackelberg Equilibrium (SE) (Stackelberg, 1934; Leitmann,

1978) by means of Monte Carlo Tree Search (MCTS) (Browne et al., 2012) simulations. A particular MCTS variant used in Mixed-UCT is Upper Confidence Bound applied to Trees (UCT) (Kocsis and Szepesvári, 2006). Mixed-UCT extends our previous approach, initially sketched in (Karwowski and Mańdziuk, 2016) and its main advantage is very good scalability, beyond state-of-the-art exact approaches. Application of Mixed-UCT leads to optimal or close-to-optimal solutions for game instances intractable by the existing algorithms.

1.1. Related literature

The most popular realization of SGs are Security Games which encompass a broad spectrum of SE-related evader-defender situations. Vast majority of Security Games presented in the literature differ from our approach by considering only single-step game scenarios. Other, more complex, multi-step SGs refer to specific (different from ours) game settings or game structure (Korzhyk et al., 2011; Johnson et al., 2012; Fang et al., 2013; Gan et al., 2015; Basilico et al., 2012). Furthermore, some patrolling games do not assume the possibility of exploiting the defender's strategy by the evader (and consequently do not aim at approximation of SE), e.g. (Zoroa et al., 2012; Saksena, 1979; Baykal-Gürsoy et al., 2014; Hohzaki and Maehara, 2010).

The most common approach to calculation of SE relies on Linear Programming (LP) (Conitzer and Sandholm, 2006) or Mixed Integer Linear Programming (MILP) (Paruchuri et al., 2008a). A typical (MI)LP approach requires the use of at least $O(nm)$ variables and $O(m)$ constraints, where n and m are the numbers of defender's and evader's strategies, respectively. Finding SE in multi-step games was proven to be NP-hard (Conitzer and Sandholm, 2006). Values of n and m grow exponentially with the number of steps making calculation of SE prohibitive. Mitigation of this problem is an important part of SG research and includes:

- the use of marginal strategies (Schlenker et al., 2016) and compact game forms, e.g. flow graphs (Wang et al., 2018), which impose additional constraints on games, most notably the requirement of being zero-sum;
- the use of column-generation strategy (Jain et al., 2010a) which relies on initially solving a game with small subset of defender's strategies and gradually adding the missing ones, according to some heuristic. Construction of an efficient heuristic is a complicated and game-dependent task that cannot be generalized to other types of games;
- the use of constraint generation (Bosanský et al., 2014) which gradually adds evader's strategies to a program. This approach is limited to games with specific utility structures, e.g. zero-sum ones;

- employment of approximation techniques with theoretically proven bounds on solution quality (Jain et al., 2011);
- application of memetic algorithms which rely on domain-based heuristics and offer very good time and memory scalability (Karwowski et al., 2019).

All the above-mentioned approaches lead to reduction of computational demands when searching for SE strategies but are restricted to specific games and require significant amount of work to be adequately applied to another type of game.

On the contrary, the approach proposed in this paper is general and straightforwardly applicable to various SG settings. The heart of the method is MCTS algorithm (Browne et al., 2012) which proved to be successful in various application domains, e.g. playing games with large state spaces (Silver and Veness, 2010; Silver et al., 2017), General Game Playing (Finnsson and Björnsson, 2008; Świechowski and Mańdziuk, 2014), planning and scheduling (Walędzik and Mańdziuk, 2018), or bus traffic regulation (Cazenave et al., 2009).

Another field of study that aims at solving problems of similar type is network interdiction (Smith et al., 2013). A typical network interdiction problem is defined on a graph with certain costs of traversing each edge and, optionally, capacities of edges. Network interdiction problem models competition between the two sides: (1) – the one that tries to traverse the network while minimizing the sum of weights of traversed arcs or maximize a flow through a network under some cost constraints and (2) – its opponent that tries to minimize the outcome of (1) by blocking some arcs in the network. The defender can play the role of either (1) or (2), depending on a particular application. For example, (1) may try to smuggle some cargo trough a border and the defender (2) is blocking some of the possible routes (Morton et al., 2007). An example of the opposite setting is resilience of a power grid, where (1) tries to build a power grid capable of supplying requested amount of power to each node and (2) attempts to destroy certain connections in the grid (Brown et al., 2006). Typically, network interdiction problems are instances of bi-level optimization, either min-max or max-min. Network interdiction covers many application areas, including interdicting development of nuclear weapons by adversary countries (Brown et al., 2009), preventing smuggling of radioactive material (Morton et al., 2007), detecting entities crossing borders or trespassing restricted areas (Romich et al., 2015), or fighting large fires, both wildfires as well as fires purposely started by a human (Rashidi et al., 2018b,a).

A model considered in this paper differs from a typical network interdiction approach in the following four main aspects:

- the defender controls one or more units which constantly move on a

graph instead of statically blocking particular arcs (Smith et al., 2013; Romich et al., 2015);

- the attacker initially possesses full knowledge of both a graph structure and the defender's strategy since the defender plays the role of the leader in Stackelberg Game model. This situation differs significantly from recent network interdiction approaches where the attacker starts with incomplete knowledge and learns new information with subsequent game plays (e.g. Borrero et al., 2016, 2018). Furthermore, in our model the defender commits to a mixed strategy instead of a deterministic assignment of patrolling routes;

- the game is general-sum and therefore each of the players maximizes their own utility function, what leads to a max-max bi-level optimization problem, formally defined in Section 2.2. Typical network interdiction games are formulated as max-min or min-max bi-level problems.

1.2. Main contribution

The main contribution of this paper is twofold:

- it presents a new method of solving *multi-step* SGs which relies on Monte Carlo Tree Search. The proposed approach is called Mixed-UCT and extends our initial works in this area (Karwowski and Mańdziuk, 2016, 2017) into a fully-developed, fast and efficient method of estimation of the optimal defender's strategy (approximation of SE) in multi-step, non-zero sum SGs and is not limited to particular game;

- it proposes a practically-motivated new type of patrolling game scenario which may be applied, for instance, to industrial areas, e.g. large factory premises;

In addition, a generator of games embedded in industrial settings[1] is presented.

To the best of our knowledge there is only one other approach to SGs that utilizes MCTS (Marecki et al., 2012) which, however, differs from ours in two significant aspects: (1) the cited method is of online nature, and (2) its goal is to optimize the sum of payoffs of a series of games in which the defender gradually discovers preferences of the evader. Consequently, the approach to calculation of mixed strategies proposed by Marecki et al. is completely different from ours.

[1]The source code of industrial-type game generator and of Mixed-UCT method is available at the project website (Mańdziuk, 2018).

The remainder of this paper is organized as follows: Section 2 formally defines a game model used in the paper. The next section introduces Mixed-UCT approach — a novel method of approximating the optimal defender's patrolling strategy in multi-step SGs. Section 4 presents experimental setup (including a generator of random industrial-type benchmark games) and experimental results. The last section provides conclusions based on experimental evaluation and specifies our future plans related to continuation of this research.

2. Game Model

The game is played by two players: the evader and the defender, on a graph $G = (V, E)$ consisting of a set of vertices (also called nodes) V and a set of undirected edges $E \subseteq \{\{x, y\} | x, y \in V \wedge x \neq y\}$. Without loss of generality we will assume that elements of V are labeled by natural numbers from 0 to $|V| - 1$. The evader is a single entity and the defender is a centralized authority that has one or more mobile units at their disposal. Both the evader and the defender's units can move between the neighboring vertices.

An underlying graph represents a spatial arrangement of industrial premises which require protection. Each vertex represents a particular part of the premises, e.g. a building, a room, a corridor, a segment of a road, etc. Edges denote direct connections between the neighboring areas. A *direct* connection means that leaving one area is equivalent to entering the other one (with no space "in between"). In the case of buildings, such connections are realized by means of gates, doors, etc. In the case of outside spaces, edges refer to adjacent areas not separated by walls, fences, etc.

Each graph $G = (V, E)$ contains several special vertices: $v_d \in V$ — a starting vertex for the defender's units, $v_e \in V$ — an entry vertex for the evader, and a few vertices in which assets are located — these vertices are referred to as *target vertices* or simply *targets*.

Each target vertex v_i has two values assigned:

- U_i^{E+}, which is the evader's reward for successfully reaching target v_i (and capturing the asset located therein)
- U_i^{D-}, which is a penalty for the defender for a successful opponent's attack (and losing the asset located in v_i).

Furthermore, each vertex v_i (either target or non-target) has assigned a reward value for the defender when he/she intercepts the evader in a given vertex, and a respective penalty for the evader. These payoffs are denoted by U_i^{E-} and U_i^{D+}, respectively. While this is not a formal requirement, typically utility values with minus sign in an upper index are negative (the respective player looses a game) and utilities with plus sign are positive

(they are assigned to the game winner). The game is *general-sum*, i.e. it is not assumed that utilities of players sum up to 0 and each player maximizes his/her own utility function.

2.1. Game rules

The game consists of at most ($T \geq 1$) rounds and is played according to the following rules:

1. A defender has n units, all initially placed in v_d; an evader has one unit initially placed in v_e;
2. In each round players simultaneously make decision about new locations of their units. Each unit can be either left in the current vertex or moved to an adjacent vertex;
3. After making the above decisions, units are placed in the chosen vertices;
4. Let's assume that the evader is currently in vertex v_k. Then,
   (a) if there is at least one defender unit in v_k, the evader is intercepted and the game ends with the payoffs U_k^{E-}, U_k^{D+} for the evader and the defender, respectively;
   (b) if v_k is a target vertex and the condition from point 4a is not met, the evader succeeds and the game ends with the respective payoffs of U_k^{E+}, U_k^{D-} for the evader and the defender;
   (c) if the limit T on the number of rounds is reached, the game ends with the payoff U_n (neutral) granted to each of the players;
   (d) otherwise the game continues from point 2.

Consequently, the game is played until either a defender's unit and the evader meet in any of the vertices, or the evader reaches a target, or the round limit T is exceeded. In the game setting, it is assumed that players cannot directly observe each other during the game. They get to know the opponent's location only when both parties meet in a common vertex (cases 4a, 4b on the above list) what ends the game.

From the perspective of the game theory, the game has *an extensive form*, is sequential (*multi-step*), of *simultaneous moves*, and with *imperfect information* (players cannot directly observe each other during the game).

2.2. Stackelberg Game

Before introducing an equilibrium of the game we describe game-theoretic idea of *pure* and *mixed* strategies. A *pure strategy*, sometimes called a *pure realization plan*, is an assignment of an action to each potentially reachable state of the game. In our game a *pure strategy* of each player is a sequence of legal decisions concerning locations of their units in each of T game rounds. Therefore, a pure strategy can be represented as a vector of vertices $\sigma \in V^T$ such that each subsequent position of a unit is either equal to or adjacent

to its previous position in the graph. In case of the defender there may be more than one unit, so in such a situation a vector of tuples is used — one element per unit. A set of all pure strategies of player $i \in \{D, E\}$ (defender and evader, respectively) will be denoted by Σ_i. Please note that some of decision points may not materialize if game ends earlier, so not all of planned decisions will be used in a particular game realization.

A *mixed strategy* is a probability distribution over Σ_i. Let's denote a set of all possible mixed strategies by Π_i and its elements by $\pi_i \in \Pi_i$. We will extend the payoff related notation $U_k^i, i \in \{E^+, E^-, D^+, D^-\}$, introduced in the previous subsection, to functions $U^i(\pi_D, \pi_E)$ which will represent the expected utility value for player i assuming that the defender commits to playing strategy π_D and the evader decides to play strategy π_E. The goal of the defender is to find a mixed strategy that maximizes their expected payoff when playing infinitely many games instead of optimizing an outcome of a single game.

Furthermore, it is assumed that once the defender commits to a particular mixed strategy, the evader can observe the defender's patrolling units for a sufficiently long time to gain knowledge about his/her mixed strategy, before deciding about an attack. In order to model this information asymmetry between the players, our model follows the principles of Stackelberg Game. Finding a solution of the game is therefore equivalent to finding the Stackelberg Equilibrium. In SG there are two players: the *leader* and the *follower*,

- the *leader* commits to some probability distribution of possible moves at the start of the game and subsequently chooses moves according to that distribution;
- the *follower* chooses their move distribution so as to maximize their expected reward, being aware of the probability distribution of the *leader's* moves.

In our model the defender plays the role of the *leader* and the evader that of the *follower*. Please note that even though the evader knows the probability distribution of the patrolling routes of defender's units, he/she does not possess precise knowledge about which route will be actually used in a given playout.

Using the above notation the SE can formally be defined as a pair (π_D, π_E) satisfying the following equations:

$$BR(\pi_D) = \arg\max_{\pi_E \in \Pi_E} U^E(\pi_D, \pi_E) \tag{1}$$

$$\arg\max_{\pi_D \in \Pi_D} U^D(\pi_D, BR(\pi_D)) \tag{2}$$

Equation (2) selects the defender's mixed strategy that yields the best utility for the defender against the best evader's response (to this strategy). The

best response is defined in equation (1) and denotes the evader's strategy that optimizes their utility against a given defender's mixed strategy.

SE is not well defined when there exists more than one best response strategy (1). For this reason, a variant of SE called Strong Stackelberg Equilibrium (SSE) (Leitmann, 1978) was proposed. SSE specifies the definition of BR by selecting the evader's response that provides the highest utility for the defender, among those fulfilling (1). Less formally, in SSE, ties among $BR(\pi_D)$ fulfilling (1) are broken in favor of the defender. The goal of the game is to find the mixed strategy for the defender, i.e a probability distribution of defender's moves, that fulfils the SSE conditions.

SE/SSE has an important theoretical property which has strong practical (computational) consequences. Namely, it can be proven (Conitzer and Sandholm, 2006) that in SE/SSE among all possible evader's best response strategies (1), there always exists at least one strategy that is in the form of a pure strategy. It is therefore possible to enumerate a *finite* number of pure evader's strategies when calculating SE/SSE. Please note that this property does not hold for the defender's strategies.

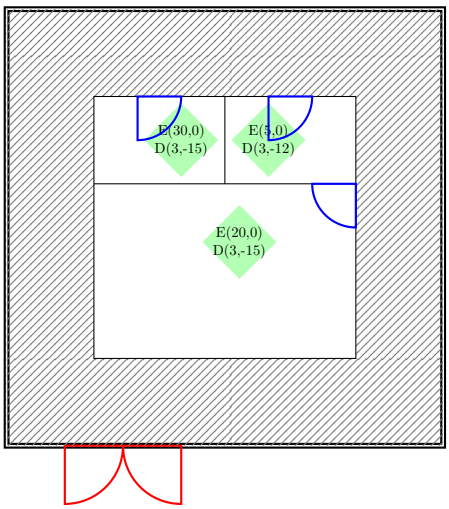
2.3. Example game

An example game is presented in Fig. 1a (industrial warehouse scenario) and Fig. 1b (its graph-based model). For simplicity, it is assumed that there is only one defender's unit and that the game has 5 steps. In order to enter any of the three target rooms (vertices 0, 1, 2) the evader, who starts in vertex 8, must go around the building. Therefore, he/she has to choose a direction, i.e. to go either clockwise or counter-clockwise. Due to round limit ($T = 5$) while going clockwise only rooms 1 and 2 can be reached by them, and when going counter-clockwise only rooms 0 and 1 are reachable.

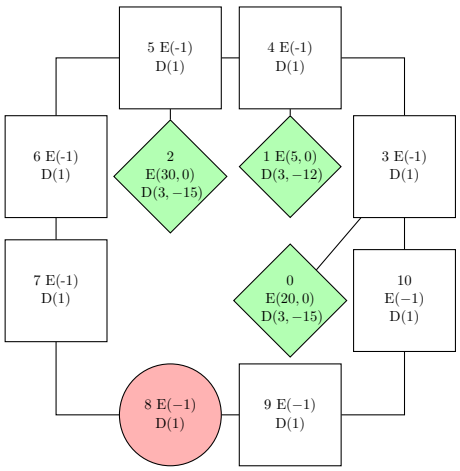
2.3.1. Strong Stackelberg Equilibrium of the game

Since in our example there is only one defending unit, pure strategies for both the evader and the defender are represented as vectors of length $T = 5$ (c.f. Section 2.2). Table 1 presents mixed strategies that comprise SSE for the example game along with the respective players' utility values. The structure of the SSE illustrates a crucial balance between two conflicting defender's goals: protecting either the clockwise path or the counter-clockwise path to the warehouse. The mixed strategy of the evader is composed of three possible sequences – three variants of a clockwise route, each of them to be chosen with probability $\frac{1}{3}$. The numbers in each sequence denote vertices visited by the evader in consecutive steps of the game. The defender's mixed strategy comprises of two move sequences with probabilities 0.4 and 0.6, respectively. Please note that the evader can use any of these three pure strategies (with probability 1), to attain exactly the same result.

Observe that at first sight playing the first sequence by the defender does not seem to be correct as the expected utility is low (equals -15, since



(a) A sketch of a warehouse. Dashed area is a fenced pavement around the building, with the only gate in the bottom left. The building consists of three rooms, all accessible from the outside, each containing an asset.



(b) Graph-based representation of the premises presented in Fig. 1a. A red circle (vertex number 8) denotes the evader entry point, diamond-shaped vertices (0, 1 and 2) are targets that contain assets. All defender's units start in vertex 0.

Figure 1: Industrial premises in the form of a warehouse building with a road around it. Each non-target vertex has two utility values assigned: $E(U_i^{E-})$ and $D(U_i^{D+})$. In target vertices there are four values: $E(U_i^{E+}, U_i^{E-})$ and $D(U_i^{D+}, U_i^{D-})$. Please see Section 2.1 for the meaning of these payoffs.

the evader successfully reaches vertex 2 in this scenario). However, playing this sequence with a certain probability (0.4) is actually indispensable for keeping the evader away from vertex 0. If this sequence was played with lower probability, the evader would switch their focus to vertex 0, as this scenario would then become much more appealing to them. This situation is illustrated in Table 2 which presents a slightly distorted defender's strategy for this game and the responsive evader's strategy. The expected payoffs for the defender and the evader are equal to -4.20 and 11.40 in the case of the optimal defender's strategy, and -7.98 and 11.81 in the case of the distorted strategy.

The following two remarks are apparent conclusions from the above example.

Remark 1. *Even a slight divergence from the optimal defender's strategy in SSE may cause significant deterioration of his/her expected utility.*

Remark 2. *Probabilities of defender's moves in the SSE strategy are often not proportional to their respective utilities under the optimal evader's response. The defender may have to play some moves with poor utility to prevent the evader from changing their strategy.*

Table 1: Optimal defender's strategy and the corresponding optimal evader's strategy for an example game depicted in Fig.1b. The rightmost columns present the expected game reward associated with the particular sequence of moves. See description within the text for a detailed discussion.

Optimal Defender			Evader		
Pr.	Sequence	$E(U_D)$	Pr.	Sequence	$E(U_E)$
0.4	0, 0, 0, 0, 0	-15	0.33	7, 7, 6, 5, 2	11.40
0.6	3, 4, 5, 2, 2	3	0.33	7, 6, 6, 5, 2	11.40
			0.33	8, 7, 6, 5, 2	11.40

Table 2: A slightly distorted optimal defender's strategy and the respective optimal evader's response strategy. Observe that even small deviation from the optimal defender's strategy may cause a complete redefinition of the evader's mixed strategy (choosing a counter clockwise path instead of a clockwise one, as in Table 1), which is now more beneficial for them.

Suboptimal Defender			Evader		
Pr.	Sequence	$E(U_D)$	Pr.	Sequence	$E(U_E)$
0.39	0, 0, 0, 0, 0	3	0.33	9, 9, 10, 3, 0	11.81
0.61	3, 4, 5, 2, 2	-15	0.33	9, 10, 10, 3, 0	11.81
			0.33	8, 9, 10, 3, 0	11.81

Both of the above remarks are critical factors taken into account when designing the method of SSE approximation presented in the next section.

3. Mixed-UCT Method

In this section the proposed UCT-based method for approximating the mixed defender's strategy in imperfect-information multi-step SG is presented. In Section 3.1, the vanilla UCT algorithm, which performs heuristic search for the best move in perfect-information games is described. This baseline UCT description is followed in Section 3.2 by a presentation of certain modifications which make the method suitable for imperfect-information games. In the last section our iterative Mixed-UCT method which relies on applying this imperfect-information UCT version against gradually improving evader, in order to derive an approximation of the optimal defender's mixed strategy is introduced.

3.1. UCT

Upper Confidence Bounds applied to Trees (UCT) proposed by Kocsis and Szepesvári (2006) is a variant of Monte-Carlo Tree Search (MCTS) method for choosing the best move for the currently moving player (UCT-player) in finite multi-step perfect-information games. UCT gradually builds a game tree composed of nodes – representing game states and edges –

representing moves/actions. Game states can be either terminal or non-terminal. For each of the players, each terminal state has some payoff value assigned. The goal is to maximize the expected long-term payoff of the UCT-player. Within the allotted time (e.g. a tournament time for making a move) UCT performs as many game simulations as possible, starting from the current game state and going down the game tree until the terminal state is reached. In that state the payoff of the UCT-player is read out and back-propagated along the move sequence which led to this state, up to the root node. Based on this back-propagated payoff value, the expected payoff (estimated quality) of each move on the path is updated.

The underlying difference between the baseline MCTS formulation and UCT is the way of choosing the next move in the above-described simulations. MCTS selects the move uniformly among all available moves, while UCT chooses the move based on the current estimation of the candidate moves' strengths and visit counters, relying on the so-called Upper Confidence Bounds (UCB) method by Auer et al. (2002). The UCB method provides a formula to balance the frequency of using the strongest moves recognized so far and the exploration of other, potentially underestimated, moves. The method addresses the classical *exploration-exploitation dilemma* (Ishii et al., 2002), offering theoretically justified optimal solution in the sense of maximization of the expected long-term cumulative payoff.

In the realm of game playing, the UCT player performs multiple simulations from the current game state. In each simulation, while in game state s being faced with the selection of move to be played (simulated), it applies the following UCB action-selection rule:

1. choose uniformly any move among those not yet tried in previous simulations (if one exists);
2. otherwise (if all moves available in a given state have been tried already), select move a^* which maximizes the following formula:

$$a^* = \arg\max_{a \in A(s)} \left\{ Q(s,a) + C\sqrt{\frac{\ln N(s)}{N(s,a)}} \right\}, \tag{3}$$

where $A(s)$ is the set of all available moves in state s, $Q(s,a)$ is the average utility of playing action a in state s in previous simulations, $N(s)$ is the number of times state s has been visited, and $N(s,a)$ is the number of times move a has been sampled in this state so far.

Formula (3) postulates to select actions with the highest expected rewards, but, on the other hand, repetitive sampling of the same moves needs to be avoided, due to incrementing of $N(s,a)$ after each trial of action a. Please observe that constant C plays a crucial role in maintaining the appropriate balance between exploration (the right term) and exploitation (the left term).

The UCT method, while simulating game actions, traverses the game tree and in each visited node chooses the move according to the UCB formula (3). As stated above, once the terminal node is reached, the accomplished utility value is propagated back, all the way up to the root node. In each node s on this path the visit counters $N(s)$ and $N(s,a)$ related to visiting state s and playing action a in that state, respectively, are incremented. Likewise, the average utility, which corresponds to quality estimation of move a in node s - $Q(s,a)$, is updated.

Once the time allotted for simulations is utilized and a *real* decision in the current root state s needs to be made, a move with the highest $Q(s,a)$ estimation is selected to be executed. Alternatively, a move with the highest number of visits (the most reliable selection) $N(s,a)$ or some combination of these two scores may be considered (Browne et al., 2012).

UCT was successfully applied to many board games, in particular Go (Silver et al., 2017; Wang and Gelly, 2007; Gelly and Silver, 2011), Havannah (Teytaud and Teytaud, 2010) or the so-called General Game Playing competitions (Finnsson and Björnsson, 2008; Świechowski et al., 2016; Świechowski and Mańdziuk, 2014; Walędzik and Mańdziuk, 2014). In all these highly challenging AI domains UCT defines *de facto* state-of-the-art approaches, as vast majority of top machine players implement this technique as its base search engine. It is important to note that in two-player games, as the number of game simulations approaches infinity, the UCT outcome converges to the min-max assessment of moves (Kocsis and Szepesvári, 2006).

Recently, the method also gained popularity in stochastic problem domains including decision problems based on Partially Observable Markov Decision Processes (Kolobov et al., 2012; Keller and Eyerich, 2012; Feldman and Domshlak, 2014), Dynamic Vehicle Routing Problems (Mańdziuk and Świechowski, 2017) or Risk-Aware Project Scheduling (Walędzik and Mańdziuk, 2018). For a detailed description of the UCT method please consult, for instance, (Kocsis and Szepesvári, 2006; Browne et al., 2012; Świechowski et al., 2015).

3.2. UCT in imperfect-information games

As stated in the previous section, UCT is typically applied to decision problems with perfect-information, whose decision sequences can be represented as paths in a graph (usually a tree or directed acyclic graph). Utilization of UCT in imperfect-information domains requires its specific adjustment. Such a modification of UCT, rendering the method suitable for imperfect-information games, called I2-UCT, was proposed in our previous work presenting the initial attempt to solving SGs (Karwowski and Mańdziuk, 2015).

In imperfect-information games each player can observe only some projection of the game state which does not comprise full game state descrip-

tion, i.e. a player can observe an *information set* containing several potentially achievable game states, which are indistinguishable for them. Information sets are used in game theory to represent imperfect information in a game (Kuhn, 1950; Basar and Olsder, 1998). In the case of our model the imperfect-information nature of the game is reflected by the fact that players are not aware of positions of the opponent's units. Consequently, the game state observed by the defender can, in principle, correspond to various real (perfect-information) game states depending on particular placement of the evader. Since the UCT-based game simulation is performed by the defender from the concrete game state, some assumption about the evader's current position must be made. This problem was addressed in (Karwowski and Mańdziuk, 2015) and, in short, resolved in the following way. In the case of imperfect-information games we propose that UCT be assisted with a certain state sampling procedure, which would yield one of the real game states (among all feasible ones) based on a sequence of defender's moves that led to a given information set. Hence, the UCT simulations do not start from a fixed current state, but instead, before each simulation, the current state (the root for the simulation) is sampled in the following way (cf. Algorithm 1 for the details):

1. Define *state* as the initial game state.
2. Play a sequence of defender's moves, which leads to the current information set, against the sampled sequence of evader's moves.

Next, a standard UCT simulation is performed from the state reached in point 2 above.

Algorithm 1: I2-UCT — Imperfect-Information UCT

Input: depth— (depth in game tree) number of previous moves
moves— vector of defender's moves that led to the current situation

```
1 for i ← 1 ... simCount do
2     state ← InitialState
3     for d ← 1 ··· depth − 1 do
4         MakeMove(state, moves [d]) // Make move and simulate
              evader response
5     SingleRun(state)// A single UCT simulation from sampled
          state.
```

The above described imperfect-information version of UCT is a core part of a method for mixed strategy approximation introduced in the next section.

Please observe that the main difference between I2-UCT and other related approaches to imperfect-information games (PIMC (Ginsberg, 2001) or ISMCTS (Cowling et al., 2012)) is that our method samples states with

respect to the opponent's strategy while the two other ones use uniform distribution for state sampling.

Another related idea is the HyperPlay algorithm proposed by Schofield et al. (2012) which also offers an interesting approach to sampling a game state after a given sequence of moves. The method differs from I2-UCT in the two following main aspects: (1) it maintains a set of possible states and refines it along with the game advancement, as opposed to sampling brand new states in each step, performed by I2-UCT; (2) it does not focus on the opponent's strategy in any specific way.

3.3. UCT for mixed strategies

The remainder of this section describes the main contribution of the paper which is an iterative algorithm for finding mixed strategy for the defender in SG. The main idea of the method was briefly announced in a short 2-page paper (Karwowski and Mańdziuk, 2016) in three possible variants. In this paper its most promising version called *Full Tree* is further developed, fully described and thoroughly evaluated. Generally speaking, within allotted time, the system performs as many game simulations as possible and gathers information about the respective evader's responses before committing to a particular mixed strategy in the real play.

Observe that in information-asymmetric (IA) games, in which the evader knows the defender's strategy, using a pure strategy by the defender is clearly a weak approach as the evader may easily exploit its deterministic nature. Instead, in IA games, the defender must adopt a mixed strategy to optimize their outcome. In other words we are facing the following bi-level optimization problem: find the defender's strategy which maximizes their payoff against the optimal evader, where the optimal evader does not use fixed strategy but a strategy which is a function of the defender's strategy.

The underpinning idea of the solution method consists of two elements:

- Use I2-UCT to gather information about effective defender's strategies played against various evaders.
- Combine the collected pure strategies into one coherent mixed strategy taking into account two remarks from Section 2.3.1.

It is worth to underline that the method does not require any specific data structures or additional resources compared to baseline UCT version. The same state-transition (game tree) structure is used to perform simulations and to collect game statistics in both versions.

There is, however, a fundamental difference on the operational level. In a typical UCT approach to sequential multi-step games, after each simulation phase a move pointed by the algorithm is played and the next UCT simulation phase starts from a new state defined by the performed move and

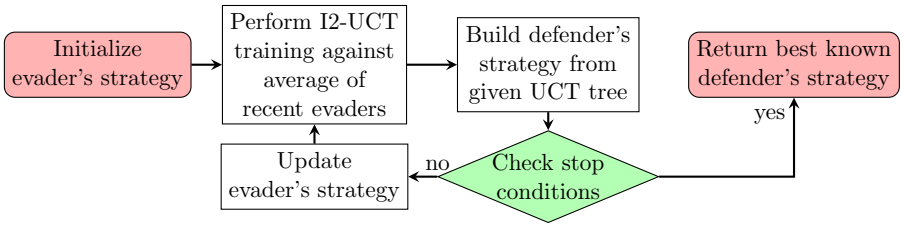


Figure 2: Outline of Mixed-UCT

an opponent's response move. In Mixed-UCT, since we are using the Stackelberg Game model, the output of the method is not a single move, but a probability distribution of move sequences (pure strategies) of length equal to the number of game rounds. In other words, in Mixed-UCT the whole simulation process is performed offline and only the final mixed strategy (a set of pure strategies of length T with assigned probabilities) is used to play the game.

Certainly, the most desirable approach to derive the final mixed strategy would be to simulate the defender player against the optimal evader in a one-shot procedure instead of making iterative simulations against gradually changing evaders. However, due to bi-level problem formulation such an approach is not feasible. Therefore, the proposed method adopts an iterative approach (see Fig. 2). The initial evader's strategy is assumed to be uniform. Then, the first UCT simulation phase (iteration) is performed with the I2-UCT method against this initial evader. Based on the outcomes, the first defender's strategy is extracted from the UCT tree (the method of strategy extraction is presented in the following subsection). Afterwards, each subsequent UCT iteration is performed against a combination of the evader's strategies developed in previous iterations. In response, in each iteration, the evader which, by game definition, is aware of the defender's mixed strategy and has the ability to explore its potential weaknesses, adapts its strategy accordingly. This iterative procedure is repeated until predefined stopping conditions are met.

The outline of the Mixed-UCT method is presented in Algorithm 2. In line 2, the initial evader's strategy is defined as the optimal one against the uniformly playing defender. In line 6, I2-UCT training is performed against the newly re-calculated evader's strategy. The evader's strategy is calculated as a uniform distribution of m most recent evader's best strategies, what proven to be more effective than using the currently optimal evader's strategy alone. I2-UCT starts with learning the first move, then learns subsequent ones (one at a time) with gradually increasing *depth* parameter (cf. a loop in line 5). This way, the training loop generates a sequence of moves of length T.

Except for the first iteration which starts off with an empty UCT tree,

Algorithm 2: Outline of the Mixed-UCT method

output: defender's strategy – the best of strategies developed in all iterations

```
1  tree ← ∅
2  evaders ←[InitES()] // Vector of evader's strategies in subsequent
       iterations initialized with the optimal strategy against uniform
       defender.
3  while not EndConditions() do
4  |   moves ← []
5  |   for depth ← 1...T  do
6  |   |   tree ← I2Uct(tree,EvaderStrategy(evaders), depth,
   |   |    moves)
7  |   |   moves ← Append(moves, BestMove(tree, depth))
8  |   defender ← StrategyFromTree(tree)
9  |   evaders ← Append(evaders,OptEvader(defender))
   // Return the best strategy among all calculated in the loop
```

each subsequent iteration begins with a tree developed in the previous iteration, which is extended by adding statistics gathered in the current iteration. The details of the way the evader's strategy is calculated in line 6 are presented in Section 3.5.

In line 8, a mixed strategy is extracted from the final UCT tree using the method described in Section 3.4. In line 9, the optimal evader's strategy is built against the currently calculated defender's strategy and appended to the list of all evader's strategies used so far. Once the above iterative algorithm is completed the best of all defender's strategies is returned. The criterion used for defender's strategy evaluation is the expected payoff obtained when playing against the optimal evader calculated in response to this strategy. Choosing the best strategy instead of the strategy developed in the last iteration is motivated by Remark 1 from Section 2.3.1.

3.4. Strategy extraction from the UCT tree

As stated in Remark 2 it is crucial for the defender to maintain a balance between moves which give them high rewards and those that prevent the evader from changing their strategy. Therefore, when building defender's strategy, the move quality estimation $Q(s, a)$ is generally not equivalent to the move playing probability. Consequently, our method does not rely on $Q(s, a)$ in the move probability estimation. Instead, a cumulative visit counter $N_{sum}(s, a)$, calculated *across all previous iterations*, is used. Hence, in the mth iteration

$$N_{sum}(s, a) = \Sigma_{i=1}^{m} N_i(s, a) \tag{4}$$

A function that calculates mixed defender's strategy from UCT tree is presented in Algorithm 3. Please note that technically we consider a single-act game and therefore each move that could be played is actually a move sequence represented as a path from the root node to a terminal game state. A loop in line 3 iterates over all such sequences.

Strategy extraction is performed in two steps. Firstly, each move sequence is assigned a weight which is equal to the product of visit counters of all single moves belonging to that sequence. For example, if the considered compound move is a sequence $A = (a_1, a_2, \ldots, a_k), k \leq T$, leading though nodes $(n_0, n_1, \ldots, n_k)$, where n_0 is a root node and n_k is a leaf node in the UCT tree, then the weight $w(A)$ for move A will be calculated as follows (cf. line 4):

$$w(A) = \Pi_{i=1}^{k} N_{sum}(n_{i-1}, a_i) \tag{5}$$

Secondly, all weights are normalized to form a proper probability distribution (line 5). Note, however, that the above-described strategy extraction procedure leads to inclusion of quite a large number of moves with low probabilities in the final probability distribution. Many of these moves were visited "occasionally" during simulations, due to exploration component pressure in eq. (3). In order to address this issue the strategy obtained in the above-mentioned way is filtered, i.e. moves with probability occurrence lower than a pre-defined threshold (α) are removed (line 6). The remaining probabilities are re-normalized to sum up to 1.

A final selection of moves with assigned probabilities comprise the mixed defender's strategy extracted from the UCT tree. As mentioned in the previous section such a strategy is extracted after each UCT iteration and represents the currently optimal strategy for the defender (cf. line 8 of Algorithm 2).

Algorithm 3: Extraction of mixed defender's strategy from the UCT tree

```
1 Function StrategyFromTree(tree)
2     moves ← ∅
3     foreach p ∈ Paths(Root(tree), TerminalState(tree)) do
4         moves ← moves ∪(p, ∏_{e∈Edges(p)} NumVisits(e))
5     moves ← NormalizeWeights(moves)
6     moves ← FilterMoves(moves,α)
7     moves ← NormalizeWeights(moves)
8     return moves
```

3.5. Building evader's strategy

Given a defender's strategy, the respective mixed strategy for the evader can be easily defined according to Algorithm 4. In the first step only com-

Algorithm 4: Calculating optimal evader's strategy against a given defender's strategy in SG

1 $M \leftarrow$ `AllMoves()`
2 $M_1 \leftarrow \{m \in M | \texttt{EvaderPayoff}(m) = \max_{m \in M} \texttt{EvaderPayoff}(m)\}$
3 $M_2 \leftarrow \{m \in M_1 | \texttt{DefenderPayoff}(m) = \max_{m \in M_1} \texttt{DefenderPayoff}(m)\}$
4 **return** `UniformStrategy`(M_2)

pound moves that provide the maximum payoff for the evader are chosen (line 2). Then, this set of selected moves (M_1) is filtered in line 3, so as to break ties in favor of the defender (cf. SSE definition in Section 2.2). In effect, only those moves from M_1 which give the maximal payoff for the defender are selected to form the set M_2, which contains the collection of optimal evader's pure strategies. Finally, any distribution of moves from M_2 defines the optimal evader's mixed strategy against a given defender's strategy. In our approach the uniform distribution is used (line 4).

4. Experimental Evaluation

In this section experimental setup and outcomes of experiments aimed at performance evaluation of Mixed-UCT method with respect to *solution quality* and *scalability* are presented. Firstly, two exact MILP-based methods, which serve as reference points for Mixed-UCT assessment, are briefly introduced. Secondly, a set of industrial-type random games is summarized in Section 4.3 (the source code of the game generator is available at the project website (Mańdziuk, 2018)). Next, the Mixed-UCT parametrization is discussed, followed by a presentation of solutions for a set of benchmark games of various sizes developed using the above-mentioned game generator. Then, the *quality of Mixed-UCT solutions* is evaluated in comparison with (exact) MILP solutions on a subset of small game instances that are feasible for MILP application. Lastly, the *scalability of Mixed-UCT* with respect to increasing game length (T) is assessed.

4.1. First reference method — DOBSS

A classical MILP formulation based on DOBSS method proposed by Paruchuri et al. (2008b) was used as the first reference method. This particular MILP formulation was chosen for several reasons. Firstly, it calculates the exact (optimal) solution and therefore serves as an excellent benchmark for quality assessment. Secondly, DOBSS method of MILP formulation was previously used as a reference approach by many authors, e.g. (An et al., 2013; Jain et al., 2010b; Kiekintveld et al., 2009; Pita et al., 2009; Shieh et al., 2012; Tsai et al., 2010).

Since DOBSS accepts single-act games only, multi-step games considered in this study had to be transformed into single-step matrix games. To this end, instead of making a single move in each turn, each player selects a compound move composed of a sequence of T moves on their first turn. Once both players choose their compound moves the game is played according to the rules presented before and completed when either the T-move sequences are played or other termination condition is met.

An optimization problem formulation for the MILP solver, adapted from DOBSS (Paruchuri et al., 2008b), is presented in eq. (6) where X is a set of all defender's moves, Q is a set of all evader's moves, R_{ij} and C_{ij} are game outcomes for the defender and the evader when the defender plays move i and the evader plays move j, respectively, and M is a large constant. Decision variables z_{ij} denote probabilities of an event that the defender plays move i and the evader plays move j, respectively, in game equilibrium, a is a slack variable, and q_j are auxiliary variables.

$$
\begin{array}{rlll}
\max_{q,z,a} \sum_{i\in X} \sum_{j\in Q} R_{ij} z_{ij} & & & \\
\text{s.t.} \sum_{i\in X} \sum_{j\in Q} z_{ij} = 1, & z_{ij} \in [0,1] & & \\
\sum_{j\in Q} z_{i,j} \le 1, & \forall_{i\in X}, & q_j \in \{0,1\} & (6) \\
q_j \le \sum_{i\in X} z_{ij} \le 1, \ \sum_{j\in Q} q_j = 1, & \forall_{j\in Q} & a \in \mathbb{R} & \\
0 \le (a - \sum_{i\in X} C_{ij} (\sum_{h\in Q} z_{ih})) \le (1-q_j)M \ , & \forall_{j\in Q} & &
\end{array}
$$

A general idea of problem specification in DOBSS is to introduce variables z_{ij} which represent joint probabilities that the defender and the evader will play strategies i and j, respectively. Additionally, the evader is restricted to choose exactly one strategy $j \in Q$, by means of auxiliary variables q_j. The last constraint in (6) forces the selected strategy j to be the best possible evader's response. A detailed description of the DOBSS method is presented in (Paruchuri et al., 2008b).

4.2. Second reference method — BC2015

As DOBSS was designed for single-step games, it does not exploit a structure of multi-step games and thus demands more computational resources than dedicated methods. Therefore, we have also compared Mixed-UCT with a method devoted to multi-act games proposed by Bosanský and Cermak (2015), referred to as BC2015 (authors' initials and publication year). The method is designed for non-zero-sum multi-step SGs and thanks to its generic formulation can be applied to various game models.

BC2015 transforms an extensive form game into its equivalent sequence form representation. Such a transformation makes the resulting MILP smaller than in the case of DOBSS, but still time and memory exponential with respect to the game length. The BC2015 program formulation adapted for our game is presented in eq. (7).

$$
\begin{aligned}
&\max_{p,r,v,s} \sum_{z \in Z} p(z) U^D(z), \text{ s.t.} \\
&v_{\inf_E(\sigma_E)} = s_{\sigma_E} + \sum_{I' \in \mathcal{I}_\mathcal{E} | seq_E(I') = \sigma_E} v_{I'} \quad + \sum_{\sigma_D \in \Sigma_D} r_D(\sigma_D) U^E(\sigma_D, \sigma_E) \quad \forall \sigma_E \in \Sigma_E \\
&r_i(\emptyset) = 1 \quad \forall i \in N \\
&r_i(\sigma_i) = \sum_{a \in A_i(I_i)} r_i(\sigma_i a) \quad \forall i \in N \quad \forall I_i \in \mathcal{I}_i, \sigma_i = seq_i(I_i) \\
&0 \leq s_{\sigma_E} \leq (1 - r_E(\sigma_E)) \cdot M \quad \forall \sigma_E \in \Sigma_E \\
&0 \leq p(z) \leq r_i(seq_i(z)) \quad \forall i \in N \quad \forall z \in Z \\
&1 = \sum_{z \in Z} p(z) \\
&r_E(\sigma_E) \in \{0, 1\} \quad \forall \sigma_E \in \Sigma_E \\
&0 \leq r_D(\sigma_D) \leq 1 \quad \forall \sigma_D \in \Sigma_D
\end{aligned}
\tag{7}
$$

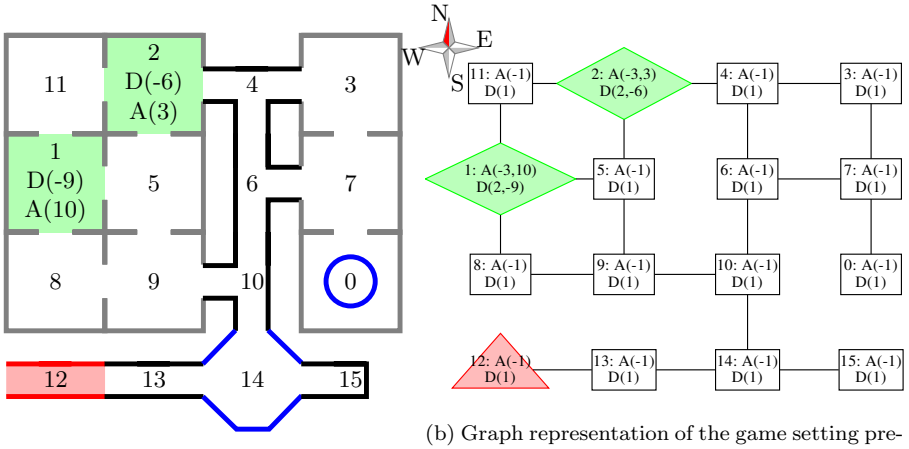
In the above equation, $N = \{D, E\}$ is a set of players, Z is a set of all possible game realizations (defined by pairs of the defender's and the evader's pure strategies), $\mathcal{I}_i$ is a set of *information sets* of player $i, i \in N$. $inf_i(\sigma_i)$ is an information set in which the last action of sequence σ_i was played. $seq_i(I)$ is a sequence of moves of player i that led to information set I.

Game transformation to MILP form in BC2015 relies on introduction of a family of variables r_i, which model the flow of probabilities of moves, i.e. for each node in a game tree and each sequence of moves, the probability that this node belongs to that sequence is equal to the sum of the respective probabilities for all its child nodes. Such a formulation is generally more effective than DOBSS. Please consult (Bosanský and Cermak, 2015) for a detailed description.

4.3. Game instances

We developed a *game generator* that creates graphs on a rectangular grid of $n \times m$ vertices which mimic the layout of a warehouse building. Each generated graph consists of several corridors, one entrance, and storage rooms, a few of which are targets. A game generation procedure is geared by a number of parameters, which are summarized in Table 3.

Once a warehouse building structure is designed, positions of valuable assets and the defender's units base are placed in random vertices within the storage space. The evader's entry point is always located at the vertex representing the building entrance. An example generated game is presented in Fig. 3. Please note that a distinction between corridors and storage rooms is not relevant for the game rules and is added only to present the warehouse generation algorithm.



(a) Structural sketch of the game.

(b) Graph representation of the game setting presented in the left figure.

Figure 3: Left: An example of a *warehouse-like* game generated with the parameter set presented in Table 3. Nodes in a 4×4 grid are numbered from 0 to 15. Narrow spaces with black outline (e.g. 6, 8, 10, 15) represent corridors. Squares with gray outlines denote storage rooms. A gap in line placed between any two vertices denotes the fact that these nodes are connected. Octagonal blue node (14) depicts a corridor crossing. A red shaded corridor space (12) denotes the evader's entry point (the building entrance), green shaded storage spaces (1 and 2) denote targets and are additionally labeled with the payoff values. Defender's units starting base (0) is denoted by a circle around the vertex number. Right: A graph representation of the above-described game setting. Target vertices (1 and 2) have additional payoff values assigned, related to the case of a successful attack.

4.3.1. Payoff structure

The model structure of the payoffs in considered SGs was presented in section 2.1 and included the reward U_i^{D+} and the penalty U_i^{E-} for the defender and the evader, respectively, in the case of catching the evader in vertex v_i. In our benchmark games one modification has been introduced, namely, the evader's penalty and the defender's reward in the case the evader is caught do not depend on a particular vertex in which this interception took place, but only on a type of this vertex, i.e. whether or not it is a target vertex. In other words catching the evader in any non-target vertex results in the same reward (penalty) for the defender (evader), regardless of a particular vertex. Likewise, a similar rule applies in the case of catching the evader in any of the target vertices. This assumption seems to be reasonable from both psychological and practical points of view. Additionally, the reward (penalty) for the defender (evader) is slightly higher in the target vertices.

A utility for the evader for successfully reaching the target located in vertex v_i (U_i^{E+}) is chosen uniformly from the range $[0, E_{max}]$, independently for each target. Similarly, a penalty for the defender in case of successful evader's attack in a given target (U_i^{D-}) is chosen uniformly from the range

Table 3: Summary of *warehouse-like* game generator parameters

parameter	value	meaning
n	4	Width of the building
m	4	Length of the building
c	1	Number of corridor crossings
p_d	0.4	Probability of placing a door from a corridor to a room
p_s	0.5	Probability of a connection between the neighboring storage rooms
U_i^{E-}	-1	Evader's penalty in a non-target vertex i
	-3	Evader's penalty in a target vertex i
E_{max}	20	Upper limit on the evader's reward
U_i^{D+}	1	Defender's reward in a non-target vertex i
	2	Defender's reward in a target vertex i
D_{max}	20	Upper limit on the defender's penalty
U_n	0	The neutral payoff (when the limit T is reached)

$[-D_{max}, 0]$, independently for each target.

4.3.2. smallbuilding *data set*

Since the game complexity grows exponentially with T, exact MILP solutions could be calculated only for games with $T \leq 6$. Therefore, a set of test games of size 4×4 with one defender's unit (which still offer vast possibility for non-trivial interactions between players within a 6-round game span) was randomly generated in the following way. First, 100 random games were generated. Next, trivial instances were removed from this set, i.e. games for which an obvious solution existed: the evader was not able to reach any target in a 6-round limit, or the evader's entry point was too close to one of the targets and the defender in no way was able to intercept them. The remaining 23 instances were used in tests. For space savings we do not present the layouts of these finally selected games. They are all available at the project website (Mańdziuk, 2018).

4.4. Mixed-UCT parameters

Steering parameters of Mixed-UCT are presented in Table 4. Their values were set based on some number of preliminary tests conducted on another set of games, mentioned in (Karwowski and Mańdziuk, 2016). Even though the currently used benchmark set differs significantly from that of (Karwowski and Mańdziuk, 2016), parameters were no further tuned in any way. The maximum number of iterations and the maximum number of iterations without improvement serve as stopping conditions for the method. They were selected high enough to ensure that no more improvement should

Table 4: Parameter set used in the experiments

Value	Parameter
40	UCT expansion coefficient (C) in UCB1 formula (3)
500	Number of UCT simulations per each tree level in one iteration (*simCount* in I2-UCT)
45 000	Maximum number of iterations
10 000	Maximum number of iterations without improvement
0.05	Filtering probability threshold (α)
500	Number of averaged previous evader's strategies (m)

Table 5: Running times of the Mixed-UCT method averaged over all test games with a given number of rounds $T = 3, \ldots, 9$, and the respective MILP results (DOBSS and BC2015) within the MILP solver feasible range of T.

num. of rounds (T)	3	4	5	6	7	8	9
Mixed-UCT time [s]	453	955	2240	2908	5233	10360	20827
DOBSS time [s]	0.03	7.33	994	n/a	n/a	n/a	n/a
BC2015 time [s]	0.20	2.20	80	9654	n/a	n/a	n/a

be expected in a given run. The filtering threshold (α) and coefficient C were selected by limited trial-and-error approach. The chosen number of simulations performed at each tree level in I2-UCT algorithm is a compromise between recalculating the evader's strategy frequently (and choosing sufficiently large number of historical evader's strategies) and an overfitting caused by training too long against the same evader.

4.5. Experimental setup and results

All Mixed-UCT experiments were run on a PC with Intel Core i7 CPU @3.40GHz and 8GB RAM running Debian Linux. The method was implemented in Java 8 technology. Experiments employing MILP approaches were run on the same machine, to allow a direct time comparison. Both MILPs were solved by Gurobi (Gurobi Optimization, Inc., 2018).

4.5.1. Quality of results

The results of Mixed-UCT are presented in reference to exact solutions of MILP and those of a uniform strategy. Using these two strategies as upper and lower bounds for Mixed-UCT outcomes, the following *relative score* measure was defined:

$$Score = (U^D_{Mixed\text{-}UCT} - U^D_{Uniform})/(U^D_{MILP} - U^D_{Uniform}) \tag{8}$$

where $U^D_{Mixed\text{-}UCT}$ and $U^D_{Uniform}$ denote the best defender's payoffs (out of 15 repetitions) obtained by Mixed-UCT and the *uniform* strategy, respectively,

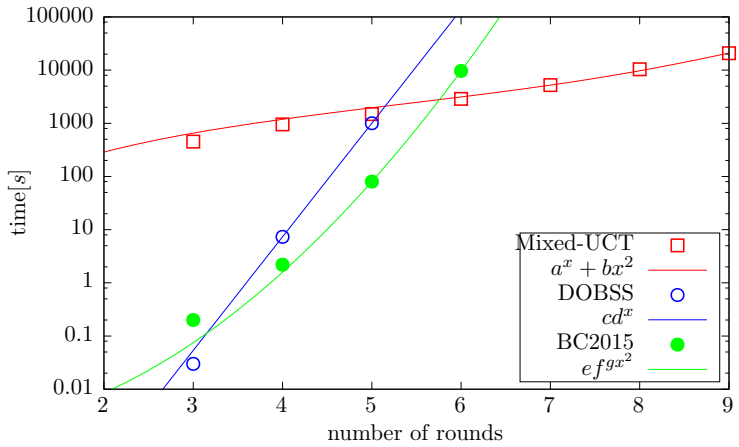


Figure 4: Running times of the Mixed-UCT method averaged over all test games (as a function of the number of rounds $T = 3, \ldots, 9$) with the approximate fitting curve, compared with the time required for calculation of the DOBSS solution (for $T = 3, 4, 5$) and BC2015 ($T = 3, \ldots, 6$). See Table 5 for exact numerical values.

and U^D_{MILP} represents the optimal defender's payoff calculated by MILP. *Score* in (8) represents the relative (standardized) quality of a Mixed-UCT result and allows making comparisons among various game settings on a common ground.

Table 6 presents the average outcomes of Mixed-UCT for $T = 5$. Results for $T = 6$ for BC2015 and Mixed-UCT are presented in Table 7. DOBSS results are not listed as their computation demands exceeded capacity of the designated hardware. None of the exact methods were able to calculate solutions beyond $T = 6$. Outcomes for $T = 3, 4$ are omitted since Mixed-UCT solved all these games perfectly.

4.5.2. Computational times

For each game, experiments for all values of $T \in \{3, 4, 5, 6, 7, 8, 9\}$ were performed, and their running times are presented in Table 5. For each T, the results are averaged over 15 repetitions per game (345 runs in total). In subsequent rows, the respective running times required for calculation of DOBSS solution (for $T = 3, 4, 5$) and BC2015 (for $T = 3, 4, 5, 6$) are presented. The same data is visualized in Fig. 4, fitted with the approximation curves. Due to extensive computational requirements, the exact solutions could not be calculated respectively for $T \geq 6$ (DOBSS) and $T \geq 7$ (BC2015).

4.6. Discussion

The quality of results was assessed using *Score* measure, shown in Table 6, which places the Mixed-UCT solution on a $[0, 1]$ scale between uniform defender's performance (value 0) and theoretically optimal solution (value

Table 6: Numerical results for all tested games, for $T = 5$. Columns from left to right present: a game number, then the average payoff, the best payoff, standard deviation, and the average running time of Mixed-UCT, then the expected payoff of a uniform defender (with uniform strategy), then theoretically best result (calculated by any of the MILP methods), the running times of DOBSS and BC2015, and the *Score* of Mixed-UCT in reference to uniform and MILP solutions, calculated according to eq. (8).

	Mixed-UCT				Uniform	Exact	DOBSS	BC2015	
Game	mean U^D	best U^D	sd	t [s]	U^D	U^D	t [s]	t [s]	Score
17	0.26	0.33	0.02	1705	-7.95	0.37	409	64	0.99
2	0	0	0	5259	-15.08	0	594	95	1
20	1.28	1.29	0.01	2200	-1.28	1.29	2705	63	1
24	1.13	1.52	0.17	2431	0.08	1.55	693	16	0.98
3	0.46	0.5	0.05	2111	-8.55	0.5	1684	69	1
35	0.47	0.5	0.03	1117	-0.89	0.5	35	25	1
39	0.04	0.06	0.01	1376	-17.06	0.09	204	32	1
41	-2.89	-2.88	0.01	1661	-5.07	-2.85	3213	35	0.99
42	0	0	0	2149	-16.53	0	1861	425	1
43	0	0	0	1236	-17.4	0	362	29	1
56	1.13	1.37	0.06	4410	0.52	1.6	1107	37	0.79
59	0.55	0.57	0.01	1632	-7.39	0.62	631	110	0.99
64	0.08	0.11	0.01	1793	-11.81	0.16	542	30	1
7	-0.79	-0.77	0.05	1744	-9.7	-0.76	2465	163	1
74	1.39	1.46	0.02	3379	-0.09	1.47	88	2	0.99
78	0.34	0.47	0.07	1380	-9.28	0.5	2464	88	1
82	0	0	0	1820	-10.66	0	899	197	1
85	-0.98	-0.9	0.02	3833	-1.79	-0.89	1532	138	0.99
87	0.77	0.78	0	3396	-1.66	0.8	9	7	0.99
89	0.02	0.21	0.05	1151	-13.78	0.21	173	14	1
91	-5.65	-5.62	0.09	2096	-5.97	-5.62	118	101	1
96	0	0	0	1409	-10.15	0.19	83	14	0.98
mean	-0.11	-0.05	0.04	2240	-7.8	-0.01	994	80	0.99

1). This relative measure allows for a direct comparison of results between games with different utility structures.

In majority of 5-step games (22 of 23) the *Score* measure exceeded 0.95 which means that the expected payoff was equal to or was very close to the optimal value. The only exception was game 56 with $Score = 0.79$. In case of $T = 6$ the results are still very promising with only two games scored below 0.95 (number 24 and 56).

Besides high quality solutions, the main asset of Mixed-UCT is its *time scalability* with respect to T. In terms of time complexity the two main contributing factors in Mixed-UCT are UCT simulation phase and calculation of an optimal evader's response. Mixed-UCT iteration cost grows quadratically with the number of rounds, since in each simulation the whole game is played and the total number of simulations grows linearly with the number of rounds (the loop in line 5 of Algorithm 2). The cost of finding the optimal evader is proportional to the number of possible evader's strategies and therefore exponential with respect to the number of game rounds. Based on this analysis, parameters a, b of the function $f(x) = a^x + bx^2$ were calculated

Table 7: Results for MixedUCT and BC2015 for 6-step games. See caption of Table 6 for columns' description.

	Mixed-UCT				Uniform	Exact	BC2015	
Game	mean U^D	best U^D	sd	t [s]	U^D	U^D	t [s]	Score
17	0.3	0.31	0.01	2238	-7.64	0.4	4134	0.99
2	0	0	0	5761	-15.05	0	7109	1
20	1.29	1.29	0	3722	-1.37	1.29	2287	1
24	1.07	1.41	0.25	4087	-0.03	1.55	28944	0.91
3	-2.18	-2.12	0.11	3304	-8.55	-2.12	7018	1
31	0	0	0	3299	-6.98	0	632	1
35	0.45	0.48	0.04	1972	-0.88	0.5	3734	0.99
39	0	0.01	0	2019	-16.99	0.14	1941	0.99
41	-2.91	-2.9	0	2846	-5.09	-2.85	1938	0.98
42	-1.6	0	0.02	1736	-16.55	0.05	71573	1
43	0	0	0	1906	-17.33	0.08	4118	1
56	0.96	1.11	0.08	4357	0.49	1.6	3821	0.56
59	0.51	0.53	0.02	1954	-8.27	0.63	11738	0.99
64	0.02	0.05	0.02	2433	-6.26	0.18	2436	0.98
7	-0.77	-0.77	0	3631	-9.72	-0.75	6590	1
74	1.47	1.47	0	2396	-0.09	1.47	1018	1
78	0.13	0.5	0.3	2791	-9.31	0.5	5779	1
82	0.58	0.6	0.01	3142	-2.58	0.62	31122	1
85	-0.97	-0.89	0.07	2169	-1.82	-0.89	11217	1
87	0.69	0.8	0.07	4943	-1.61	0.8	479	1
89	0.04	0.19	0.09	2078	-13.75	0.25	1417	1
91	-5.62	-5.62	0	1741	-5.97	-5.62	12149	1
96	0.17	0.19	0.01	2360	-9.89	0.19	849	1
mean	-0.29	-0.23	0.05	2908	-7.18	-0.09	9654	0.97

using nonlinear least-squares Levenberg-Marquardt algorithm (Levenberg, 1944), leading to $a = 2.92$ and $b = 68.929$.

In terms of computation times, Mixed-UCT was inferior to MILP methods for small games ($T \leq 5$). In case of $T = 6$, running times of Mixed-UCT are approximately 3 times shorter than those of BC2015 and significantly shorter than the *estimated* DOBSS results. Projection of BC2015 running times presented in Figure 4 shows that Mixed-UCT is orders of magnitude faster for $T = 7, 8, 9$ than both MILP approaches.

5. Conclusions and Future Work

This paper introduces a new model of patrolling game which is played on a graph representing spatial arrangement of an area under protection and assumes *information asymmetry* between the evader and the defender. The existence of IA is attributed to the fact that, in practice, the evader can usually observe the defender's units patrolling schedules prior to performing an attack. To model the game we employ Stackelberg Game principles and present a novel approach to approximation of Stackelberg Equilibrium, called Mixed-UCT, which relies on Monte Carlo Tree Search. The quality of Mixed-UCT solutions is experimentally assessed on a set of randomly

generated non-trivial games which are played in warehouse-like, industrial environment.

Experimental results show that Mixed-UCT is able to efficiently approximate Stackelberg Equilibrium for moderate size games. The results presented in Tables 6 and 7 prove that solutions obtained for games of 5 or 6 steps are optimal or close to optimal in vast majority of the cases. Out of 23 non-trivial test games only two received a weaker *Score* value, below 0.95. A deeper analysis of these games (24 and 56) revealed that in both of them a set of defender's strategies contains at least one strategy that has the following three properties: (1) there exist an evader's strategy, paying against which leads to high defender's payoff; (2) playing against the evader's optimal response strategy results in low defender's payoff; (3) the equilibrium strategy profile yields moderate defender's payoffs against all possible evader's strategies. Since Mixed-UCT tends to prefer strategies that yield high utility against at least some evader's strategies rather than those with moderate payoffs against all evader's strategies, strategy (1) will be preferred over equilibrium strategy (3). Resilience of Mixed-UCT to such structures is one of our research goals for the near future.

In terms of running time requirements the method is orders of magnitude faster than the exact MILP-based approaches for games of length beyond 6 steps. In fact, for such games the computation time requirements of DOBSS and BC2015 are unacceptable from the practical application viewpoint.

Currently we work on extension of the model and adequate modification of Mixed-UCT in the two main directions. The first one refers to *observability* of the opponent. In the current model players cannot see each other unless they meet in the same vertex. In many practical settings it would be desirable to assume that players can see each other from some distance, without immediate capturing, e.g. through windows or fences, and adjust their decision strategies accordingly based on these observations. Besides extending our model and gradually improving Mixed-UCT, another interesting research avenue is to design a family of UCT-based methods or a unified generic framework that will be applicable to settings typical for network interdiction problems.

The other direction is related to *bounded rationality* of the evader's decisions. In real-life situations players (people), due to limited observability of the opponent's strategies, stress or other distractive factors, often do not make perfectly rational decisions. In this context we plan to employ two realizations of bounded rationality: *quantal response equilibrium* model (McKelvey and Palfrey, 1995), in which the resultant strategy of the evader is defined as a certain distortion of their optimal strategy, and *limited observability of the defender's strategy*, where the evader calculates the optimal response strategy but against distorted (not optimal) defender's strategy.

Acknowledgments

The authors would like to thank anonymous reviewers for their insightful comments. This work was supported by the National Science Centre, Poland, grant number 2017/25/B/ST6/02061.

References

An, B., Ordóñez, F., Tambe, M., Shieh, E., Yang, R., Baldwin, C., DiRenzo III, J., Moretti, K., Maule, B., and Meyer, G. (2013). A deployed quantal response-based patrol planning system for the US Coast Guard. *Interfaces*, 43(5):400–420.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Basar, T. and Olsder, G. J. (1998). *Dynamic Noncooperative Game Theory, 2nd Edition*. Society for Industrial and Applied Mathematics.

Basilico, N., Gatti, N., and Amigoni, F. (2012). Patrolling Security Games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial Intelligence*, 184:78–123.

Baykal-Gürsoy, M., Duan, Z., Poor, H. V., and Garnaev, A. (2014). Infrastructure security games. *European Journal of Operational Research*, 239(2):469–478.

Borrero, J. S., Prokopyev, O. A., and Sauré, D. (2016). Sequential shortest path interdiction with incomplete information. *Decision Analysis*, 13(1):68–98.

Borrero, J. S., Prokopyev, O. A., and Sauré, D. (2018). Sequential interdiction with incomplete information and learning. *Operations Research (Accepted)*.

Bosanský, B. and Cermak, J. (2015). Sequence-form algorithm for computing Stackelberg Equilibria in extensive-form games. In Bonet, B. and Koenig, S., editors, *Procedings of AAAI*, pages 805–811.

Bosanský, B., Kiekintveld, C., Lisý, V., and Pechoucek, M. (2014). An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *JAIR*, 51:829–866.

Brown, G., Carlyle, M., Salmerón, J., and Wood, K. (2006). Defending critical infrastructure. *Interfaces*, 36(6):530–544.

Brown, G. G., Carlyle, W. M., Harney, R. C., Skroch, E. M., and Wood, R. K. (2009). Interdicting a nuclear-weapons project. *Operations Research*, 57(4):866—-877.

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

Cazenave, T., Balbo, F., and Pinson, S. (2009). Using a Monte-Carlo approach for bus regulation. *Procedings of International IEEE Conference on Intelligent Transportation Systems*, pages 340–345.

Conitzer, V. and Sandholm, T. (2006). Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90. ACM.

Cowling, P. I., Powley, E. J., and Whitehouse, D. (2012). Information Set Monte Carlo Tree Search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):120–143.

Fang, F., Jiang, A. X., and Tambe, M. (2013). Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *Proceedings of AAMAS*, pages 957–964.

Feldman, Z. and Domshlak, C. (2014). On Monte-Carlo Tree Search: To MC or to DP? In *Proceedings of ECAI*, pages 321–326.

Finnsson, H. and Björnsson, Y. (2008). Simulation-based approach to General Game Playing. In *Proceedings of AAAI*, pages 259–264.

Gan, J., An, B., and Vorobeychik, Y. (2015). Security games with protection externalities. In *Proceedings of AAAI*, pages 914–920.

Gelly, S. and Silver, D. (2011). Monte-Carlo Tree Search and Rapid Action Value Estimation in Computer Go. *Artificial Intelligence*, 175(11):1856–1875.

Ginsberg, M. L. (2001). GIB: Imperfect information in a computationally challenging game. *JAIR*, 14:303–358.

Gurobi Optimization, Inc. (2018). Gurobi optimizer reference manual. `http://www.gurobi.com/documentation/8.0/`.

Haveman, J. D., Shatz, H. J., and Vilchis, E. A. (2005). U.S. port security policy after 9/11: Overview and evaluation. *Journal of Homeland Security and Emergency Management*, 2(4).

Hohzaki, R. and Maehara, H. (2010). A single-shot game of multi-period inspection. *European Journal of Operational Research*, 207(3):1410 – 1418.

Ishii, S., Yoshida, W., and Yoshimoto, J. (2002). Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural Networks*, 15(4-6):665—-687.

Jain, M., Kardes, E., Kiekintveld, C., Ordónez, F., and Tambe, M. (2010a). Security games with arbitrary schedules: A branch and price approach. In *Proceedings of AAAI*, pages 792–797.

Jain, M., Kiekintveld, C., and Tambe, M. (2011). Quality-bounded solutions for finite Bayesian Stackelberg games: scaling up. In *Proceedings of AAMAS*, pages 997–1004.

Jain, M., Tsai, J., Pita, J., Kiekintveld, C., Rathi, S., Tambe, M., and Ordóñez, F. (2010b). Software Assistants for randomized patrol planning for the LAX airport police and the Federal Air Marshal Service. *Interfaces*, 40(4):267–290.

Johnson, M. P., Fang, F., and Tambe, M. (2012). Patrol strategies to maximize pristine forest area. In *Proceedings of AAAI*, pages 295–301.

Karwowski, J. and Mańdziuk, J. (2015). A new approach to security games. In *Proceedings of ICAISC*, pages 402–411. Springer-Verlag.

Karwowski, J. and Mańdziuk, J. (2016). Mixed strategy extraction from UCT tree in security games. In *Proceedings of ECAI*, pages 1746–1747.

Karwowski, J. and Mańdziuk, J. (2017). The impact of the number of averaged attacker's strategies on the results quality in Mixed-UCT. In *Porceedings of ICAISC*, pages 477—-488. Springer International Publishing.

Karwowski, J., Mańdziuk, J., Żychowski, A., Grajek, F., and An, B. (2019). A memetic approach for sequential security games on a plane with moving targets. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19) (Accepted)*.

Keller, T. and Eyerich, P. (2012). Prost: Probabilistic planning based on UCT. In *Proceedings of ICAPS*, pages 119–127.

Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordóñez, F., and Tambe, M. (2009). Computing optimal randomized resource allocations for massive Security Games. In *Proceedings of AAMAS*, pages 689–696.

Kocsis, L. and Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer.

Kolobov, A., Mausam, and Weld, D. S. (2012). LRTDP versus UCT for online probabilistic planning. In *Proceedings of AAAI*, pages 1786–1792.

Korzhyk, D., Conitzer, V., and Parr, R. (2011). Solving Stackelberg Games with uncertain observability. In *Proceedings of AAMAS*, pages 1013–1020.

Kuhn, H. W. (1950). Extensive games. *Proceedings of the National Academy of Sciences*, 36(10):570–576.

Leitmann, G. (1978). On generalized Stackelberg strategies. *Journal of Optimization Theory and Applications*, 26(4):637—-643.

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168.

Mańdziuk, J. (2018). Simulation-based methods in multi-step Stackelberg Security Games in the context of homeland security. `https://sg.mini.pw.edu.pl/`.

Mańdziuk, J. and Świechowski, M. (2017). UCT in Capacitated Vehicle Routing Problem with Traffic Jams. *Information Sciences*, 406:42–56.

Marecki, J., Tesauro, G., and Segal, R. (2012). Playing repeated Stackelberg games with unknown opponents. In *Proc. of AAMAS*, pages 821–828.

McKelvey, R. D. and Palfrey, T. R. (1995). Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38.

Morton, D. P., Pan, F., and Saeger, K. J. (2007). Models for nuclear smuggling interdiction. *IIE Transactions*, 39(1):3—-14.

Neuringer, A. (1986). Can people behave "randomly?": The role of feedback. *Journal of Experimental psychology: general*, 115(1):62.

Paruchuri, P., Pearce, J. P., Marecki, J., Tambe, M., Ordonez, F., and Kraus, S. (2008a). Efficient algorithms to solve Bayesian Stackelberg Games for security applications. In *Proceedings of AAAI*, pages 1559–1562.

Paruchuri, P., Pearce, J. P., Marecki, J., Tambe, M., Ordonez, F., and Kraus, S. (2008b). Playing games for security: an efficient exact algorithm for solving Bayesian Stackelberg Games. In *Proceedings of AAMAS*, pages 895–902.

Pita, J., Jain, M., Ordóñez, F., Tambe, M., Kraus, S., and Magori-Cohen, R. (2009). Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *Proceedings of AAMAS*, pages 369–376.

Rashidi, E., Medal, H., and Hoskins, A. (2018a). An attacker-defender model for analyzing the vulnerability of initial attack in wildfire suppression. *Naval Research Logistics (NRL)*, 65(2):120—-134.

Rashidi, E., Medal, H. R., and Hoskins, A. (2018b). Mitigating a pyro-terror attack using fuel treatment. *IISE Transactions*, 50(6):499–511.

Romich, A., Lan, G., and Smith, J. C. (2015). Algorithms for optimizing the placement of stationary monitors. *IIE Transactions*, 47(6):556—-576.

Rubinstein, A. (1998). *Modeling bounded rationality*, volume 1. MIT press.

Saksena, J. (1979). Beat patrolling in urban areas a case study. *European Journal of Operational Research*, 3(3):199—-206.

Schlenker, A., Brown, M., Sinha, A., Tambe, M., and Mehta, R. (2016). Get me to my GATE on time: Efficiently solving general-sum bayesian threat screening games. In *Proceedings of ECAI*, pages 1476–1484.

Schofield, M., Cerexhe, T., and Thielscher, M. (2012). HyperPlay: A solution to General Game Playing with Imperfect Information. In *Proceedings of AAAI*, pages 1606–1612.

Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., and Meyer, G. (2012). PROTECT: A deployed game theoretic system to protect the ports of the United States. In *Proceedings of AAMAS*, pages 13–20.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., et al. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676):354—-359.

Silver, D. and Veness, J. (2010). Monte-Carlo planning in large POMDPs. In *Annual Conference on Neural Information Processing Systems*, pages 2164–2172.

Smith, J. C., Prince, M., and Geunes, J. (2013). Modern network interdiction problems and algorithms. pages 1949–1987. Springer New York.

Stackelberg, H. v. (1934). Marktform und Gleichgewicht. *Vienna: Springer*.

Świechowski, M. and Mańdziuk, J. (2014). Self-adaptation of playing strategies in General Game Playing. *Computational Intelligence and AI in Games, IEEE Transactions on*, 6(4):367–381.

Świechowski, M., Mańdziuk, J., and Ong, Y.-S. (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):218–228.

Świechowski, M., Park, H.-S., Mańdziuk, J., and Kim, K.-J. (2015). Recent advances in General Game Playing. *The Scientific World Journal*, 2015:Article ID: 986262.

Tambe, M. (2011). *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press.

Teytaud, F. and Teytaud, O. (2010). Creating an Upper-Confidence-Tree program for Havannah. *Advances in Computer Games*, pages 65–74.

Tsai, J., Yin, Z., Kwak, J.-y., Kempe, D., Kiekintveld, C., and Tambe, M. (2010). Urban security: Game-theoretic resource allocation in networked physical domains. In *Proceedings of AAAI*, pages 881–886.

Walędzik, K. and Mańdziuk, J. (2014). An automatically-generated evaluation function in General Game Playing. *Computational Intelligence and AI in Games, IEEE Transactions on*, 6(3):258–270.

Walędzik, K. and Mańdziuk, J. (2018). Applying hybrid Monte Carlo Tree Search methods to risk-aware project scheduling problem. *Information Sciences*, 460–461:450–468.

Wang, X., An, B., Strobel, M., and Kong, F. (2018). Catching Captain Jack: Efficient time and space dependent patrols to combat oil-siphoning in international waters. In *Proceedings of AAAI*, pages 208–215.

Wang, Y. and Gelly, S. (2007). Modifications of UCT and sequence-like simulations for Monte-Carlo Go. *CIG*, 7:175–182.

Zoroa, N., Fernández-Sáez, M. J., and Zoroa, P. (2012). Patrolling a perimeter. *European Journal of Operational Research*, 222(3):571–582.