UDC 004.056:519.832.4 DOI: 10.18698/0236-3933-2019-2-22-39

SADDLE POINT SEARCH ALGORITHM FOR THE PROBLEM OF SITE PROTECTION LEVEL ASSIGNMENT BASED ON SEARCH OF SIMPLICES' FACES ON HYPERPLANES OF EQUAL DIMENSION

A.Yu. Bykov abykov@bmstu.ru
M.V. Grishunin grishunin-mv@ya.ru
I.A. Krygin krygin.ia@gmail.com

Bauman Moscow State Technical University, Moscow, Russian Federation

Abstract

This paper deals with a continuous zero-sum game with constraints on resources between a defender allocating resources for protection of sites and an attacker choosing sites for attack. The problem is formulated so that each player would have to solve its own linear program with a fixed solution of the other player. We show that in this case the saddle point is located on the faces of simplices defining feasible solutions. We propose an algorithm of saddle point search based on search of the simplices' faces on hyperplanes of equal dimension. Each possible face is defined using a boolean vector defining states of variables and problem constraints. The search of faces is reduced to the search of feasible boolean vectors. In order to reduce computational complexity of the search we formulate the rules for removing patently unfeasible faces. Each point of a face belonging to an $(m-1)$-dimensional hyperplane is defined using m points of the hyperplane. We created an algorithm for generating these points. Two systems of linear equations must be solved in order to find the saddle point if it located on the faces of simplices belonging to hyperplanes of equal dimension. We created a generic algorithm of saddle point search on the faces located on hyperplanes of equal dimension. We present an example of solving a problem and the results of computational experiments

Keywords

Information security, game theory, zero-sum game, linear programming, saddle point, linear equations system

Received 14.05.2018
© Author(s), 2019

Introduction. We consider a problem of assigning protection level for different sites (servers, workstations, etc.) in a certain system. These sites store data of different degree of confidentiality or importance and require different protection levels. This approach can also be applied to nodes of a computational network in intrusion prevention systems; the nodes can be of greater or lesser importance. The resources for site protection are usually limited. The resources may include money, computational resources required for protection software and so on.

Game theory is often used for solving the information protection problems. As a rule, there are two players: a defender and an attacker, although other interpretations are possible. The examples of attacker and defender games are presented in [1–9]. The examples of games with multiple players are presented in [10–12]; a game with a theoretically infinite number of players (mean field games) is considered in [13].

Equilibrium states are often used as problem solution in game theory. It is a saddle point for zero-sum games and a Nash equilibrium state for games with non-opposing interests.

In this paper we consider an algorithm of saddle point search in the site protection level assignment problem. This is a zero-sum game; we use a risk-oriented approach to define the utility function as possible damage stemming from site security breach. The problem setting is similar to the ones presented in [14, 15]; but its character is more generic.

1. Statement of the problem of site protection level assignment.

1.1. Input data. *Basis sets.*

1. $Z=\{z_1, z_2, \ldots, z_m\}$ — set of sites to be protected, indexed by $M=\{1, 2, \ldots, m\}$.

2. $R=\{r_1, r_2, \ldots, r_l\}$ — set of limited protection resources indexed by $L=\{1, 2, \ldots, l\}$.

3. $N=\{n_1, n_2, \ldots, n_s\}$ — set of limited attack resources indexed by $S=\{1, 2, \ldots, s\}$.

Parameters of the elements of the sets and the relationships between them.

1. $w_i > 0, \ \forall\ i \in M$ — possible damage (site cost) if security of the i-th site is breached.

2. $p_{pr\ i} \in (0, 1), \ \forall i \in M$ — probability (possibility) of preventing an attack on the i-th site if it is protected.

3. $a_{ki} \in [0, 1), \ \forall k \in L, \ i \in M$ — normalized value of the k-th limited resource used for protecting the i-th site. The total resource value is equal to one (the resources' values can be used without normalization if necessary).

4. $b_k \in (0, 1], \ \forall k \in L$ — maximum normalized value of the k-th limited protection resource.

5. $c_{ki} \in \left[0,1\right), \forall k \in S,\ i \in M$ — normalized value of the k-th limited resource used for attacking the i-th site. The total resource value is equal to one (the resources' values can be used without normalization if necessary).

6. $d_k \in \left(0,1\right],\ \forall k \in S$ — maximum normalized value of the k-th limited attack resource.

1.2. Decision variables. We introduce a variable $p_i \in \left[0,\ 1\right], \forall i \in M$ corresponding to the site protection level (protection probability). These variables form a vector $\vec{P}$. For the attacker we introduce a variable $q_i \in \left[0,1\right],\ \forall i \in M$ corresponding to importance of attacking a site (attack probability). These variables form a vector $\vec{Q}$.

1.3. Utility functions of the players. The utility functions of the players are defined by damage to the defender. The average damage can be given by as

$$U\left(\vec{P},\ \vec{Q}\right) = U_{\max}\left(\vec{Q}\right) - U_{\text{prevent}}\left(\vec{P},\ \vec{Q}\right) = \sum_{i \in M} w_i q_i - \sum_{i \in M} p_{pr\,i} w_i p_i q_i, \quad (1)$$

where $U_{\max}\left(\vec{Q}\right) = \sum\limits_{i \in M} w_i q_i$ is maximum damage that can be done by the attacker if these is no protection; $U_{\text{prevent}}\left(\vec{P},\ \vec{Q}\right) = \sum\limits_{i \in M} p_{pr\,i} w_i\, p_i q_i$ is damage prevented by the defender.

The defender wishes to minimize the utility function, the attacker wishes to maximize it.

1.4. Constraints. The system of constraints imposed on the protection resources defining the set of feasible alternatives $\Delta_{\text{feas}}^{(P)}$ is given by

$$\Delta_{\text{feas}}^{(P)} : \left\{ \sum_{i \in M} a_{ki} p_i \le b_k,\ \forall k \in L. \right. \quad (2)$$

The system of constraint imposed on the attack resources defining the set of feasible alternatives $\Delta_{\text{feas}}^{(Q)}$ is given by

$$\Delta_{\text{feas}}^{(Q)} : \left\{ \sum_{i \in M} c_{ki} q_i \le d_k,\ \forall k \in S. \right. \quad (3)$$

We assume that the system of constraints (2) and (3) do not allow the players to choose the solutions consisting only of ones (total protection or total attack), as this this case the solutions are optimal, and the problem becomes trivial.

Thus, a linear program (LP) must be solved for finding the decision variables (unknown vectors $\vec{P}$ or $\vec{Q}$) if the other player's solution is fixed.

We consider the algorithms of saddle point search on the faces of simplices defined in the m-dimensional spaces by the systems of constraints (2) и (3).

A saddle point is a pair of vectors $\vec{P}^*$ и $\vec{Q}^*$ satisfying the following conditions

$$\begin{aligned} &U\left(\vec{P}^*, \vec{Q}^*\right) \le U\left(\vec{P}, \vec{Q}^*\right), \forall \vec{P} \in \Delta_{\text{feas}}^{(P)}; \\ &U\left(\vec{P}^*, \vec{Q}^*\right) \ge U\left(\vec{P}^*, \vec{Q}\right), \forall \vec{Q} \in \Delta_{\text{feas}}^{(Q)}. \end{aligned} \tag{4}$$

2. Algorithms of saddle point search on the faces of simplices based on search of faces. The level set method for convex-concave function saddle point search is proposed in [16] and earlier papers. This method is approximate and requires solving convex programming problems including ones with implicit objective function. It is also shown in [16] that a saddle point always exists with the given conditions. If an LP solution exists it is located on the boundary of a simplex (it can be in a vertex or there may a set of solutions located on the same face).

The saddle point defined by $\vec{P}^*$ и $\vec{Q}^*$ can be located in the vertices of the simplices defined by (2) и (3) as well as on the faces of the simplices. In this case the vertices of the simplices can be considered as a special case when using this method. We consider an algorithm of saddle point search if it is located on the faces of simplices defined by constraints (2) and (3) and conditions $p_i \in \left[0,1\right]$, $q_i \in \left[0,1\right]$, $\forall i \in M$.

The simplices (polygons of feasible solutions) are in the m-dimensional metric spaces. In this case each of the constraints (2) and (3) defines in the space an (m–1)-dimensional hyperplane if an inequality becomes an equality. The feasible solutions are located inside unit hypercubes because $p_i \in \left[0,1\right], \forall i \in M$ and $q_i \in \left[0,1\right], \forall i \in M.$ The hypercubes are bounded by (m–1)-dimensional hyperplanes given by equations $p_i = 0,\ \forall i \in M,\ p_i = 1,$ $\forall i \in M$ for the defender and $q_i = 0,\ \forall i \in M,\ q_i = 1,\ \forall i \in M$ for the attacker. Thus, the simplices are bounded by hyperplanes defined by constraints (2) и (3) (if inequalities become equalities) and hyperplanes being the bounds of the hypercubes. If possible, the intersection of two (m–1)-dimensional hyperplanes is an (m–2)-dimensional hyperplane. If possible, the intersection of three (m–1)-dimensional hyperplanes is an (m–3)-dimensional hyperplane and so on. A special point of the m-dimensional space in this case is a zero-dimensional hyperplane. The faces of the simplices can belong to hyperplanes of different dimension (the terms *face*, *edge*, *vertex* are used for in 3-dimensional case). We consider the algorithm of saddle point search on the faces of simplices based on search of faces.

2.1. Basis of the algorithm of the exhaustive search of the simplices' faces. In order to define a point on an $(m-1)$-dimensional hyperplane defining a space of the same dimension it is necessary to find m different points. These points should not simultaneously be on any of the $(m-2)$-dimensional hyperplanes belonging to the initial $(m-1)$-dimensional space. For example, three points of a plane defining two-dimensional space in the initial 3-dimensional space should not be on the same line (analogy of the points defining $(m-1)$ linearly independent vectors forming the basis and the point defining the center of coordinates).

Let $\vec{P}^{(1)}, \vec{P}^{(2)}, \ldots, \vec{P}^{(m)}$ be the points of a hyperplane a simplex face in the defender space is located in. These points may not be feasible from the standpoint of constraints (2) or the conditions $p_i \in [0, 1], \forall i \in M.$ The main condition is that all the points must be located on any single $(m-2)$-dimensional hyperplane belonging to the original $(m-1)$-dimensional space. Then any point of the $(m-1)$-dimensional hyperplane can be given by as $\vec{P} = \sum\limits_{i \in M} \alpha^{(i)} \vec{P}^{(i)},$ where coefficients $\alpha^{(i)}$ do not have to be positive; they must meet the normalization condition $\sum\limits_{i \in M} \alpha^{(i)} = 1$ (the sum should not necessary be equal to one, it can be any non-zero value). The same goes for the attacker. $\vec{Q}^{(1)}, \vec{Q}^{(2)}, \ldots, \vec{Q}^{(m)}$ are points of a hyperplane where a simplex face is located. Any point of $(m-1)$-dimensional hyperplane can be given by as $\vec{Q} = \sum\limits_{i \in M} \beta^{(i)} \vec{Q}^{(i)},$ where $\sum\limits_{i \in M} \beta^{(i)} = 1.$ We introduce the way of obtaining these points.

We will be searching the saddle point assuming that at least one of the constraints (3) on the attacker's resources is an equality, i.e., at least one limited resource is full depleted. Otherwise the attacker would be able of improving its utility function by increasing any component q_i which is not equal to one.

Similarly, we assume that the inequality becomes the equality for at least one of the constraints (2) for the defender saddle point. If the opposite is true, then some $p_i < 1$ can be possibly increased for an infinitely small value without violating the constraint. If w_i and $p_{pr\,i}, \forall i \in M$ are not equal to zero this can be done provided that $q_i = 0,$ as only in this case the defender utility will not improve, and the initial defender solution will satisfy the saddle point condition. With such an increase of p_i (or some of them) the security of the i-th site increases [15], and the attacker has no stimulus of increasing q_i; the attacker solution will still satisfy the saddle point condition. In this case the initial solution of the defender where the resource is not fully depleted can be considered a special case of the solution obtained by increasing p_i, when the resource is fully depleted.

We consider definition of hyperplanes of different dimension on the example of the defender vector space $\vec{P}$ (the attacker hyperplanes are defined in the same way). To that end we introduce an $l + m + m$-dimensional boolean vector $\overrightarrow{Pl}_{(d)}$ (the attacker vector is $\overrightarrow{Pl}_{(a)}$, the vector dimension is $s+m+m$). Components indexed by $1, \ldots, l$ correspond to the constraints (2). If a component is equal to one, the inequality in the corresponding constraint (2) becomes the equality. This constraint will be referred to as active further on. At least one of the components is going to be equal to one.

The components $l+1, \ldots, l+m$ correspond to the fixed element of the solution vector and $p_i = 1,\ i \in M$, if the component $l+i$ is equal to one. Otherwise, the element p_i is not fixed (variable). The components $l+m+1, \ldots,$ $l+m+m$ correspond to the fixed element of the solution vector and $p_i = 0$ if the component $l+m+i$ is equal to one. Otherwise, the element p_i is not fixed. Obviously, the vector elements $l+i$ и $l+m+i,\ \forall i \in M$ cannot be equal to one at the same time.

Consider the case of two constraints and three objects ($l = 2$ and $m = 3$). Two-dimensional hyperplanes are defined by the vectors where each of them has a single one: $\overrightarrow{Pl}_{(d)} = \|10000000\|$ and $\overrightarrow{Pl}_{(d)} = \|01000000\|$, as one of the constraints (2) must always be active. One-dimensional hyperplanes (lines in the 3-dimensional space), are defined by vectors, each of them has two ones:

$$\|11000000\|,\ \|10100000\|,\ \|10010000\|,\ \|10001000\|,\ \|10000100\|,$$
$$\|10000010\|,\ \|10000001\|,\ \|01100000\|,\ \|01010000\|,\ \|01001000\|,$$
$$\|01000100\|,\ \|01000010\|,\ \|01000001\|.$$

Zero-dimensional hyperplanes (a point in the 3-dimensional space) are defined by vectors having three ones. Thus, the search of hyperplane comes down to search of boolean vectors.

We need to iterate through all possible combinations of the similar vectors considering the following constraints:

– at least one of the components 1, …, l for the defender (1, …, s for the attacker) must be equal to one.

– components $l+i$ and $l+m+i$, $\forall i \in M$ should not be equal to one simultaneously (the fixed element of the vector $\vec{P}$ cannot simultaneously be equal to zero and one), the same applies for the attacker.

Search of vectors is a computationally complex task. We consider supplementary methods for removing patently unfeasible hyperplanes in order to reduce computational complexity.

2.2. Removing unfeasible hyperplanes. The rules of removing hyperplanes are formulated as follows.

1. Remove unfeasible hyperplanes defined by constraints (2) and (3). Some of the constraints in (2) and (3) may be unfeasible, i.e., the inequality in a constraint will never become the equality (the constraint will not become active) because it will not be allowed by other constraints. An LP must be solved in order to check feasibility for each of the constraints in (2) and (3).

For the defender:

$$F_k^{(P)}\left(\vec{P}\right) = \sum_{i \in M} a_{ki} p_i \to \max_{\vec{P} \in \Delta_{\text{feas}}^{(P)}}, \ \forall k \in L, \tag{5}$$

k-th constraint is feasible if the obtained solution will turn the k-th constraint inequality into the equality; otherwise, the constraint is unfeasible.

Similarly, for the attacker an LP must solved for each of the constraints (3):

$$F_k^{(Q)}\left(\vec{Q}\right) = \sum_{i \in M} c_{ki} q_i \to \max_{\vec{Q} \in \Delta_{\text{feas}}^{(Q)}}, \ \forall k \in S, \tag{6}$$

k-th constraint is feasible if the obtained solution will turn the k-th constraint inequality into the equality, otherwise the constraint is unfeasible.

Gradient vector for the objective functions (5) and (6) is normal to the hyperplanes defined by k-th constraint. If the constraint is feasible, the solution must be in the corresponding hyperplane.

The unfeasible constraint of the systems (2) and (3) can be removed.

2. Remove hyperplanes because of resource shortage. When considering vectors that define hyperplanes it is possible to separate decision variables in $\vec{P}$ and $\vec{Q}$, into two non-intersecting sets $M = M^{(\text{free})} \cup M^{(\text{fixed})}$, $M^{(\text{free})}$ indices of free variables; $M^{(\text{fixed})}$ indices of fixed variables; variables' value can be zero or one and they remain constant for all points of a hyperplane.

A defender hyperplane can be removed if $\exists\ k \in L, \sum\limits_{i \in M^{(\text{fixed})}} a_{ki} p_i > b_k$. In this case the resource constraint will be violated for each point of a hyperplane. The same can be applied for the attacker; if $\exists\ k \in S, \sum\limits_{i \in M^{(\text{fixed})}} c_{ki} q_i > c_k$ then the remove the corresponding hyperplane.

3. Removing hyperplanes because of active constraint's unfeasibility. Consider a certain component of a vector that defines a hyperplane. The vector is indexed by 1, …, l for the defender and 1, …, s for the attacker. Assume the component's values is one (the corresponding constraint in system (2) or (3) is active). We denote the index of this component with k (constraint index in the system (2) or (3)). If $\sum\limits_{i \in M^{(\text{free})}} a_{ki} < b_k - \sum\limits_{i \in M^{(\text{fixed})}} a_{ki} p_i$, then the hyperplane defined by this vector can be removed as the k-th constraint will not be active. Indeed, even if all free variables are equal to one (required resource defined in the left-hand side of the inequality) the remaining resource (right-hand side of the inequality) will not be depleted. For the attacker this condition is given by as: $\sum\limits_{i \in M^{(\text{free})}} c_{ki} < d_k - \sum\limits_{i \in M^{(\text{fixed})}} c_{ki} q_i$.

2.3. Obtaining points defining a hyperplane. It is necessary to define m points in order to define an $(m-1)$-dimensional hyperplane. These points may not satisfy the constraints (2) and (3) and they may not be contained inside a unit hypercube. All these points must not be on any of the $(m-2)$-dimensional hyperplanes being a subset of the original hyperplane. We consider two cases.

1. Consider a vector defining a hyperplane indexed by 1, …, l for the defender and 1, …, s for the attacker. Assume one its components is equal to one, the other components are equal to zero within specified ranges (an active constraint case). In this case the number of required points is equal to $card\ (M^{(\text{free})})$. Let the index of this component be k. Then the remaining k-th resource for the defender is going to be equal to $b_k^{(\text{remain})} = b_k - \sum\limits_{i \in M^{(\text{fixed})}} a_{ki} p_i$.

Then we set the values of free variables for the defender:

$$\left\| \frac{b_k^{(\text{remain})}}{a_{k1}}\ 0\ 0 \ldots 0 \right\|, \left\| 0\ \frac{b_k^{(\text{remain})}}{a_{k2}}\ 0 \ldots 0 \right\|, \ldots, \left\| 0\ 0\ 0 \ldots \frac{b_k^{(\text{remain})}}{a_{km^{(\text{free})}}} \right\|;$$

$a_{k1}, a_{k2}, ..., a_{km^{(\text{free})}}$ are indexed by the set $M^{(\text{free})}$, $m^{(\text{free})} = card\left(M^{(\text{free})}\right) \le m$. These points turn the inequality in the *k*-th constraint into the equality. We conduct the same procedure for the attacker.

2. Assume that more than one component of the vector mentioned above is equal to one. The number of such components (active constraints) is denoted with $n > 1$. The corresponding active constraints are indexed by $N = \{1, 2, \ldots, n\}$ (these constraints are indexed separately). Let $m^{(\text{free})}$ be the number of free variables in vectors $\vec{P}$ or $\vec{Q}$ (the values for the defender and the attacker may differ, but we are going to denote them in the same way in order to reduce the number of indices given that the search is conducted separately). Let $m^{(\text{fixed})}$ be the number of fixed variables in vectors $\vec{P}$ or $\vec{Q}$ (the values for the defender and the attacker may be different). Let $K = m - n - m^{(\text{fixed})} + 1$ be the number of points that define a (*K*–1)-dimensional hyperplane. Values of free variables have to be calculated in order to find each of the *K* points. Each point is calculated by solving a system of *n* linear equations (the case of active constraint, the inequality becomes the equality). We do this in the following way: we choose *n* variables from the set of free variables, the total number of combinations is $C^n_{m^{(\text{free})}}$, each combination can define one of *K* points, searching through all combinations may not be necessary. We keep the selected *n* variables and set the other free variables equal to zero. Then we formulate a system of *n* equations (the system below is shown for the defender; the similar system is formulated for the attacker):

$$\left\{ \sum_{i \in N} a_{ji}\, p_i = b_j^{(\text{remain})}, \ \forall j \in N. \right. \tag{7}$$

The variables in (7) are indexed by $N = \{1, 2, \ldots, n\}$, *n* is the number of remaining free variables. If according to Kronecker — Capelli theorem the system (7) has a unique solution, we find it and obtain a point of a hyperplane; then go to the next $m^{(\text{free})}$ choose *n* combination. If the system does not have a solution we go to the next combination. The process is repeated until all *K* points are found.

2.4. Saddle point search on the faces of simplices on the hyperplanes of equal dimension. The solution of an LP is located on the simplex face belonging to a hyperplane with dimension greater than zero (the face is not a point) if the objective function gradient vector is normal to the hyperplane. In this case the solution of the problem includes all points of the face. In the game with the utility function (1) and constraints (2) and (3) the point $\vec{P}^*$ must ensure that the utility function is maximized on the face of the second player in the vector space $\vec{Q}$ and is equal for all points on the face. Similarly, the point $\vec{Q}^*$ on the face must ensure that the utility function is minimized on the face of the first player in the vector space $\vec{P}$ and is equal for all point of the face.

We consider the saddle point search in case the simplices faces of the defender and the attacker are on the hyperplanes of equal dimension.

If each face of the defender and the attacker is on the (K–1)-dimensional hyperplane this face can be defined by K points. Let $\vec{P}^{(1)}, \vec{P}^{(2)}, \ldots, \vec{P}^{(K)}$ be the points on the face in the vector space $\vec{P}$, and $\vec{Q}^{(1)}, \vec{Q}^{(2)}, \ldots, \vec{Q}^{(K)}$ be the points on the face in the vector space $\vec{Q}$. We introduce the multipliers $\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(K)}$ and $\beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(K)}$. The saddle point is going to be search for as: $\vec{P}^* = \sum_{i=1}^{K} \alpha^{(i)} \vec{P}^{(i)}$, $\vec{Q}^* = \sum_{i=1}^{K} \beta^{(i)} \vec{Q}^{(i)}$.

The system of linear equations for the unknown multipliers $\alpha^{(1)}$, $\alpha^{(2)}, \ldots, \alpha^{(K)}$ is formulated as follows:

$$\begin{cases} U\left(\vec{P}^*, \vec{Q}^{(1)}\right) = U\left(\vec{P}^*, \vec{Q}^{(2)}\right), \\ U\left(\vec{P}^*, \vec{Q}^{(1)}\right) = U\left(\vec{P}^*, \vec{Q}^{(3)}\right), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ U\left(\vec{P}^*, \vec{Q}^{(1)}\right) = U\left(\vec{P}^*, \vec{Q}^{(K)}\right), \\ \sum_{i=1}^{K} \alpha^{(i)} = 1. \end{cases} \tag{8}$$

The system of linear equations for the unknown multipliers $\beta^{(1)}$, $\beta^{(2)}, \ldots, \beta^{(K)}$ is formulated as follows:

$$\begin{cases} U\left(\vec{P}^{(1)}, \vec{Q}^*\right) = U\left(\vec{P}^{(2)}, \vec{Q}^*\right), \\ U\left(\vec{P}^{(1)}, \vec{Q}^*\right) = U\left(\vec{P}^{(3)}, \vec{Q}^*\right), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ U\left(\vec{P}^{(1)}, \vec{Q}^*\right) = U\left(\vec{P}^{(K)}, \vec{Q}^*\right), \\ \sum_{i=1}^{K} \beta^{(i)} = 1. \end{cases} \tag{9}$$

If systems of equations (8) and (9) are consistent and there exist the unique solutions $\vec{P}^*$ and $\vec{Q}^*$ that satisfy the constraints (2) and (3) and the saddle point conditions (4), then they define a saddle point. In most cases these conditions may not be satisfied simultaneously. In order to iterate through all combinations of faces the exhaustive search of the simplices faces is required.

The example of a saddle point on the faces for the two-dimensional case is presented in Fig. 1. It is seen that the saddle point is on the faces in the space $\vec{P}$ и $\vec{Q}$, the end points of the faces (vertices) are denoted as *1* and *2* , the saddle point itself is denoted as *3*. The saddle point value in the space $\vec{P}$ ensures that the utility function gradient in the space $\vec{Q}$ is normal to the face (*1*, *2*). Similarly, the anti-gradient of the utility function in the space $\vec{P}$ is normal to the face (*1*, *2*). The gradient and anti-gradient are shows as separate lines coming from the center of coordinates. This ensures that the utility function on the faces (*1*, *2*) is constant for each player if the other player's solution is in the point *3*. Figure 1 also shows a solution in case the saddle point is searched for in the space $\vec{P}$ on the face (*1*, *2*), and in the space $\vec{Q}$ on the face (*2*, *4*). In this case the points that satisfy the systems (8) and (9) are denoted as *5*. Obviously, these points are unfeasible.

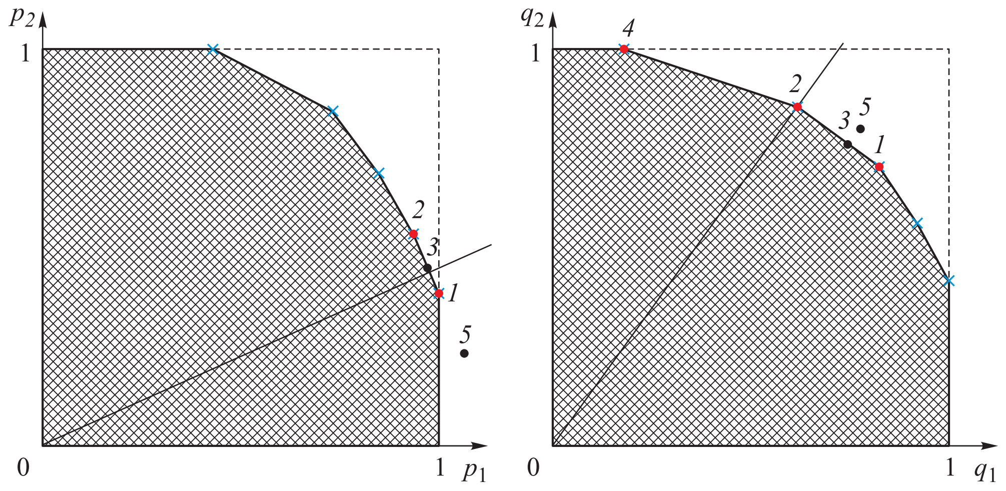

Fig. 1. An example of the two-dimensional case of a saddle point on the faces belonging to the hyperplanes of equal dimension

It is worth noting that there may be a situation when a saddle point is on the faces belonging to the hyperplanes of different dimension. An example for the two-dimensional case is shown in Fig. 2. The example is almost identical to

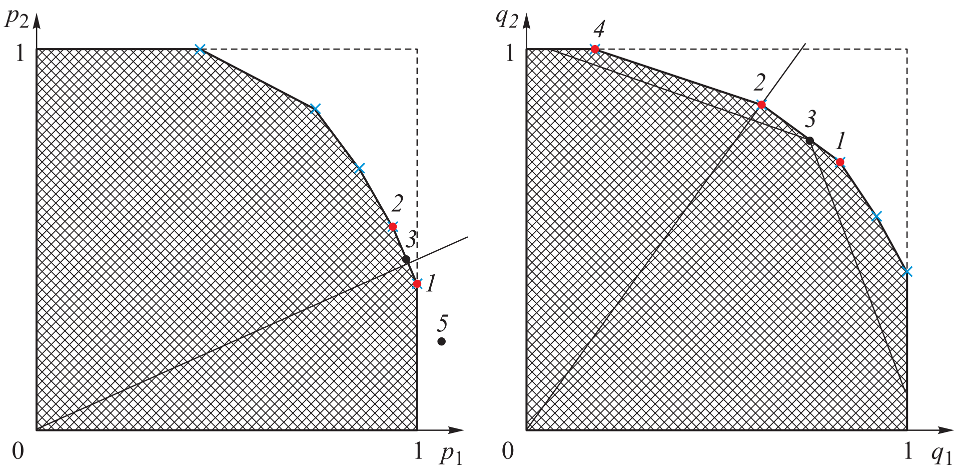

Fig. 2. A two-dimensional example of a saddle point on the faces belonging to hyperplanes of different dimension

the one shown in Fig. 1, but new constraints are introduced in the space $\vec{Q}$ (two lines coming from the point *3*). In this case the point *3* value in the space $\vec{Q}$ ensures the same values of the utility function on the face defined by points (*1*, *2*) in the space $\vec{P}$. In this case as a rule the saddle point in the space $\vec{P}$ is not going to be unique. This will be a certain set of points on the face (*1*, *2*) near the point *3*, so that the gradient direction in the space $\vec{Q}$ defined by these points would yield a solution in the point *3* for the attacker. In this example the face (*1*, *2*) in the space $\vec{P}$ is in the one-dimensional hyperplane, and the point *3* in the space $\vec{Q}$ is in the zero-dimensional hyperplane.

2.5. Algorithm of exhaustive search of faces on hyperplanes of equal dimension. We assume that the "redundant" constraints have been removed from systems (2) and (3) according to Rule 1 from Subsection 2.2 prior to starting the algorithm.

Step 0. Set $K=(m-1)$-the dimension of the considered hyperplanes. Set the list of points defining hyperplanes in the attacker space *listQ* as empty.

Step 1. Set the initial vector for the defender. The vector defines a *K*-dimensional hyperplane: $\overrightarrow{Pl}_{(d)}=\|1\ldots0\ 0\ 0\|$ (the vector initially contains only one element equal to 1, in general, the vector has (*m*–*K*) ones).

Step 2. Check feasibility of the vector $\overrightarrow{Pl}_{(d)}$ using rules 2 and 3 from Subsection 2.2. If the vector is feasible, go to Step 3; otherwise, go to Step 12.

Step 3. Find the points $\vec{P}^{(1)}, \vec{P}^{(2)}, \ldots, \vec{P}^{(K+1)}$ of the vector $\overrightarrow{Pl}_{(d)}$ that define a *K*-dimensional hyperplane using rules formulated in Subsection 2.3. If no points are found, go to Step 12.

Step 4. If *listQ* is empty, set the initial vector for the attacker. The vector defines a *K*-dimensional hyperplane $\overrightarrow{Pl}_{(a)} = \left\| 1 \ldots 0\ 0\ 0 \right\|$ (initially the vector contains only one element equal to one, in general the vector has $(m-K)$ ones). If the list is not empty, go to Step 7.

Step 5. Check feasibility of the vector $\overrightarrow{Pl}_{(a)}$ using rules 2 and 3 from Subsection 2.2. If the vector is feasible, go to next Step; otherwise, go to Step 11.

Step 6. Find points $\vec{Q}^{(1)}, \vec{Q}^{(2)}, \ldots, \vec{Q}^{(K+1)}$ for the vector $\overrightarrow{Pl}_{(a)}$, defining a *K*-dimensional hyperplane using rules formulated in Subsection 2.3. If the points are found, put the points in the list *listQ*, go to Step 8. If no points found, go to Step 11.

Step 7. Extract $\vec{Q}^{(1)}, \vec{Q}^{(2)}, \ldots, \vec{Q}^{(K+1)}$ from the first element of the list *listQ* without removing it. The list pointer goes to the next element.

Step 8. Solve two systems of equations as described in Subsection 2.4 for the points $\vec{P}^{(1)}, \vec{P}^{(2)}, \ldots, \vec{P}^{(K+1)}$ and $\vec{Q}^{(1)}, \vec{Q}^{(2)}, \ldots, \vec{Q}^{(K+1)}$. If the unique solution exists, find $\vec{P}^{*}$ and $\vec{Q}^{*}$, go to next Step; otherwise, go to Step 10.

Step 9. Check if $\vec{P}^{*}$ and $\vec{Q}^{*}$ satisfy the saddle point condition according to Subsection 2.4. If the conditions are satisfied, the saddle point has been found, terminate the algorithm; otherwise, go to next Step

Step 10. If *listQ* is ready to be used (full), extract the next element (points $\vec{Q}^{(1)}, \vec{Q}^{(2)}, \ldots, \vec{Q}^{(K+1)}$) without removing it; the pointer goes to the next element. If the pointer is not at the end of the list, go to Step 8; otherwise, go to Step 12. If the list *listQ* is not ready to be used (not full), go to next Step.

Step 11. Using the current vector defining the attacker hyperplane $\overrightarrow{Pl}_{(a)}$, calculate the next vector $\overrightarrow{Pl}_{(a)}$, defining another hyperplane of the same dimension using rules of vector search described in Subsection 2.1. If the next vector is found, go to Step 5; otherwise, set the list *listQ* as ready to be used (full), go to next Step.

Step 12. Using the current vector defining the defender hyperplane $\overrightarrow{Pl}_{(d)}$, calculate the next vector $\overrightarrow{Pl}_{(d)}$, defining the other hyperplane of the same dimension using rules of vector search described in Subsection 2.1. If this vector exists, go to Step 2. If the vector search is completed, go to next Step.

Step 13. If $K > 0$, set $K = K - 1$, clear *listQ*, go to Step 1; otherwise, stop, face search is finished, the saddle point is not found.

The list of points *listQ* defining hyperplanes is used to speed up the algorithm, so that the algorithm would not have to generate the same set of hyperplanes in the attacker space for different hyperplanes of the same dimension of the defender space.

3. Example of solving a problem. We consider an example of solving a problem with the following parameters: 8 sites, 4 limited protection resources, 1 limited attacker resource.

The sites can represent servers in some automated system. Possible damage values in case of site security breach (in conditional units) and breach prevention probabilities using protection resources are shown in Table 1.

Table 1

Site damage and prevention probability values

Damage w_i, cond. units	4000	10 000	3000	9000	5000	9500	10 000	8000
Probability $p_{\text{prevent } i}$	0.8	0.99	0.7	0.95	0.85	0.9	0.5	0.92

The limited protection resources include: protection cost (not normalized), CPU resources, RAM resources, disk memory resources. The defender resources' parameters (coefficients in the constraint system (2)) are presented in Table 2.

Table 2

Defender resource constraint system parameters

Constraint No.	Coefficient value in the left-hand side of the constraints, a_{ki}								Values of b_k
1	100	1000	200	900	400	500	1200	1100	3000
2	0.03	0.2	0.05	0.15	0.3	0.25	0.35	0.1	0.7
3	0.05	0.15	0.2	0.25	0.3	0.1	0.33	0.35	0.6
4	0.01	0.05	0.02	0.05	0.02	0.01	0.01	0.01	0.2

One constraint is imposed on the attacker — cost constraint (cost data is shown without normalization), the parameters of this resource for the attacker are shown in Table 3.

Table 3

Attacker resource constraint system parameters

Constraint No.	Coefficient values in the left-hand side of the constraints, c_{1i}								Value of d_1
1	50	600	60	500	100	120	1000	550	1500

The constraints 1 and 4 were removed according to rules formulated in Subsection 2.2 for the given input data. A saddle point on the 4-dimensional hyperplanes defined by 5 different points was found as the result of the search of the faces belonging to hyperplanes of equal dimension using the presented algorithm.

The components of the boolean vector $\overrightarrow{Pl}_{(d)}$, defining a hyperplane the solution for the defender space is in are presented in Table 4 (last row).

Table 4

Boolean vector $\overrightarrow{Pl}_{(d)}$ components, defining a hyperplane in the defender space

Constraint No.		Corresponding components of p_i are equal to one								Corresponding components of p_i are equal to zero							
2	3	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

The components of the boolean vector $\overrightarrow{Pl}_{(a)}$, defining a hyperplane the solution for the defender space is in are presented in Table 5 (last row).

Table 5

Boolean vector $\overrightarrow{Pl}_{(a)}$ components, defining a hyperplane in the attacker space

Constraint No.	Corresponding components of q_i are equal to one								Corresponding components of q_i are equal to zero							
1	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0

Component values of the vectors $\vec{P}$ and $\vec{Q}$ defining the saddle point are shown in Fig. 6. The utility function value for this solution is 19483.3 conditional units.

Table 6

Obtained saddle point solution

$\vec{P}$	1	0.404	0	0.468	0.521	0.971	0	0.340
$\vec{Q}$	1	0.215	1	0.414	1	0.166	0.564	0.674

We also conducted computational experiments to determine the frequency of finding solution when searching on hyperplanes of different dimension (sometimes there is no solution). To that end we generated the input data using pseudorandom number generators; the number of facilities was equal to 5, the number of limited resources for the defender and the attacker was equal to 4. We solved 1000 problems. In three cases out of 1000 (0.3 %) the solution on the faces belonging to hyperplanes of equal dimension was not found. In this case it is necessary to conduct search of the faces belonging to hyperplanes of different dimension which is beyond the scope of this paper. We should note that if input data was not generated using pseudorandom number generators the probability of repeating the same or close values for certain parameters in the input data would be significantly higher. In this case the probability of not finding a solution using the proposed algorithm will be higher.

Conclusion. In this paper we presented a continuous two player zero-sum game with constraints imposed on the resources defining site protection level and the choice of site to be attacked. Each player solves a linear program for the fixed solution of the other player. A saddle point exists for this problem and it is on the faces of simplices. Two systems of linear equations have to be solved in order to find the saddle point if it is located on the faces of simplices belonging to hyperplanes of equal dimensions. One system of linear equations defines the defender solution, the other — the attacker solution. We propose using boolean vector to define the hyperplanes. In this case the solution is obtained by searching pairs of hyperplanes for the defender and the attacker. This task in turn is reduced to searching pairs of boolean vectors.

The proposed approach allows finding a saddle points in most cases if the input data is generated using pseudorandom number generators. The future studies will be devoted to developing algorithms of saddle point search on the faces of simplices belonging to hyperplanes of different dimension or when at least one of the systems of equations (8) and (9) has multiple solutions.

Translated by U. Gordeeva

REFERENCES

[1] Ma C.Y.T., Yau D.K.Y., Rao N.S.V. Scalable solutions of Markov games for smart-grid infrastructure protection. *IEEE Trans. Smart Grid*, 2013, vol. 4, no. 1, pp. 47–55. DOI: 10.1109/TSG.2012.2223243

[2] Koppel A., Jakubiec F.Y., Ribeiro A. A saddle point algorithm for networked online convex optimization. *IEEE Trans. Signal Process.*, 2015, vol. 63, no. 19, pp. 5149–5164. DOI: 10.1109/TSP.2015.2449255

[3] Paramasivan B., Prakash M., Kaliappan M. Development of a secure routing protocol using game theory model in mobile ad hoc networks. *J. Commun. Inf. Netw.*, 2015, vol. 17, no. 1, pp. 75–83. DOI: 10.1109/JCN.2015.000012

[4] Wang Q., Zhu J. Optimal information security investment analyses with the consideration of the benefits of investment and using evolutionary game theory. *2nd Int. Conf. Information Management (ICIM)*, 2016, pp. 105–109. DOI: 10.1109/INFOMAN.2016.7477542

[5] Schöttle P., Böhme R. Game theory and adaptive steganography. *IEEE Trans. Inf. Forensics Security,* 2016, vol. 11, no. 4, pp. 760–773. DOI: 10.1109/TIFS.2015.2509941

[6] Lei C., Ma D., Zhang H. Optimal strategy selection for moving target defense based on Markov game. *IEEE Access*, 2017, vol. 5, pp. 156–169.
DOI: 10.1109/ACCESS.2016.2633983

[7] Xiao L., Chen T., Han G., et al. Channel-based authentication game in MIMO systems. *IEEE GLOBECOM Conf*., 2016, pp. 1–6. DOI: 10.1109/GLOCOM.2016.7841657

[8] Shah S., Chaitanya A., Sharma V. Resource allocation in fading multiple access wiretap channel via game theoretic learning. *ITA Workshop*, 2016, pp. 1–7.
DOI: 10.1109/ITA.2016.7888137

[9] Li L., Shamma J. Efficient computation of discounted asymmetric information zero-sum stochastic games. *54th IEEE Conf. CDC*, 2015, pp. 4531–4536.
DOI: 10.1109/CDC.2015.7402927

[10] Freudiger J., Manshaei M.H., Hubaux J.P., et al. Cooperative location privacy. *IEEE Trans. Depend. Sec. Comput*., 2013, vol. 10, no. 2, pp. 84–98.
DOI: 10.1109/TDSC.2012.85

[11] Gupta A., Langbort C., Başar T. Dynamic games with asymmetric information and resource constrained players with applications to security of cyberphysical systems. *IEEE Trans. Control Netw. Syst.,* 2017, vol. 4, no. 1, pp. 71–81.
DOI: 10.1109/TCNS.2016.2584183

[12] Chessa M., Grossklags J., Loiseau P. A game-theoretic study on non-monetary incentives in data analytics projects with privacy implications. *IEEE 28th Computer Security Foundations Symp*., 2015, pp. 90–104. DOI: 10.1109/CSF.2015.14

[13] Wang Ya., Yu F.R., Tang H., et al. A mean field game theoretic approach for security enhancements in mobile ad hoc networks. *IEEE Trans. Wireless Commun*., 2014, vol. 13, no. 3, pp. 1616–1627. DOI: 10.1109/TWC.2013.122313.131118

[14] Bykov A.Yu., Shmatova E.S. The algorithms of resource distribution for information security between objects of an information system based on the game model and principle of equal security of objects. *Nauka i obrazovanie: nauchnoe izdanie* [Science and Education: Scientific Publication], 2015, no. 9 (in Russ.). DOI: 10.7463/0915.0812283

[15] Bykov A.Yu., Krygin I.A., Mullin A.R. Algorithms of the protection system between assets of a mobile device on the basis of zero-sum game and equal security principle. *Vestn. Mosk. Gos. Tekh. Univ. im. N.E. Baumana, Priborostr.* [Herald of the Bauman Moscow State Tech. Univ., Instrum. Eng.], 2018, no. 2, pp. 48–68 (in Russ.).
DOI: 10.18698/0236-3933-2018-2-48-68

[16] Gol'shteyn E.G. Generalized saddle version of the level method. *Comput. Math. Math. Phys*., 2001, vol. 41, no. 8, pp. 1083–1091.

Bykov A.Yu. — Cand. Sc. (Eng.) Assoc. Professor, Department of Information Security, Bauman Moscow State Technical University (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

Grishunin M.V. — Post-Graduate Student, Department of Information Security, Bauman Moscow State Technical University (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

Krygin I.A. — Post-Graduate Student, Department of Information Security, Bauman Moscow State Technical University (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

Please cite this article as:

Bykov A.Yu., Grishunin M.V., Krygin I.A. Saddle point search algorithm for the problem of site protection level assignment based on search of simplices' faces on hyperplanes of equal dimension. *Herald of the Bauman Moscow State Technical University, Series Instrument Engineering*, 2019, no. 2, pp. 22–39.
DOI: 10.18698/0236-3933-2019-2-22-39