Iterative Computation of Security Strategies of Matrix Games with Growing Action Set

Lichun Li · Cedric Langbort

Published online: 8 September 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract
This paper studies how to efficiently update the saddle-point strategy, or security strategy of one player in a matrix game when the other player develops new actions in the game. It is well known that the saddle-point strategy of one player can be computed by solving a linear program. Developing a new action will add a new constraint to the existing LP. Therefore, our problem becomes how to efficiently solve the new LP with a new constraint. Considering the potentially huge number of constraints, which corresponds to the large size of the other player’s action set, we use the shadow vertex simplex method, whose computational complexity is lower than linear with respect to the size of the constraints, as the basis of our iterative algorithm. We first rebuild the main theorems in the shadow vertex method with a relaxed non-degeneracy assumption to make sure such a method works well in our model, then analyze the probability that the old optimum remains optimal in the new LP, and finally provide the iterative shadow vertex method whose average computational complexity is shown to be strictly less than that of the shadow vertex method. The simulation results demonstrate our main results about the probability of re-computing the optimum and the computational complexity of the iterative shadow vertex method.

Keywords Game theory · Growing action set · Iterative computation · Shadow vertex method

This work was supported in part by NSF Grants 1619339 and 1151076 (Directorate for Computer and Information Science and Engineering) to Cedric Langbort.

Lichun Li
lichunli@eng.famu.fsu.edu

Cedric Langbort
langbort@illinois.edu

1 Department of Industrial and Manufacturing Engineering, FAMU-FSU College of engineering, Tallahassee, FL 32310, USA

2 Coordinated Science Lab, University of Illinois at Urbana-Champaign, Champaign, IL 61801, USA
1 Introduction

In many noncooperative games, players can develop new actions as the game is being played. For example, in cyber-security scenarios, attackers can suddenly start employing theretofore unknown system vulnerabilities in the form of so-called zero-day attacks. The goal, then, for the defender, is to quickly and efficiently modify its response so as to reach an equilibrium of the newly formed game.

This paper focuses on two player zero-sum games where one player, say player 2, can develop new actions and has a growing action set. Player 1, who has a fixed action set, may need to update its strategy to the new saddle-point strategy when a new action is revealed. Player 1’s saddle-point strategy can be computed by solving a linear program (LP), and adding a new action is nothing more than adding a new constraint to the existing LP. Hence, our problem becomes how to solve the LP with a new constraint efficiently.

A popular approach to handle linear programs in which new constraints are introduced is to work with the dual problem, whose feasible set can only grow when constraints are added. In turn, a currently feasible dual point can be used as an initial point in the search for the optimal solution to the new dual problem, in the presence of the new constraint.

However, when the LP of interest has a large number of constraints compared to the number of variables (as is the case for the problems we consider, in which attackers may possess a large action set while defenders have relatively few options), the dual LP has a large number of variables and methods such as the dual simplex algorithm may thus prove ineffective compared to a primal problem-based approach. This is the reason why we prefer to directly solve the primal LP, and develop an iterative method for constraint-handling that reduces average computational complexity.

The corresponding dual problem looks similar to an online LP problem where new variables are introduced to the existing LP [1,7,9,11,14]. The key idea to approximate the optimum in online LP is to assign the new variable 1, if the reward is greater than the shadow price based cost, 0 otherwise. Unlike in the problems considered in this literature, however, the new variables introduced in the LP are not restricted to be boolean, and the central idea developed in these papers cannot be applied to our problem.

Online conic optimization is another online tool which looks for an approximated optimum of a convex optimization problem where new variables are introduced to the existing convex optimization [8]. A special requirement in online conic optimization is that the constraints of variable are separated. Eghbali and Fazel [8] provide an algorithm to approximate the optimum based on the form of the optimal primal-dual pair which is related to the concave conjugate of the objective function. The main issue to adopt the algorithm in our problem is that to separate the constraints of variables, we need to reformulate the objective function, and the concave conjugate of the reformulated objective function is not well defined.

While neither online LP nor online conic optimization can be directly used in our model, our previous work [3] showed how adaptive multiplicative weights could be used to find an approximated solution to the fast and efficient saddle-point strategy updating goal as well, by adding a simple condition to the Freund and Schapire scheme. In this paper, we present an algorithm to solve this updating problem exactly.

Considering that our LP may have a large number of constraints (corresponding to a large action set of player 2), we use the shadow vertex method, whose average computational complexity grows very slowly with respect to the number of constraints [4,5], as the basis for our iterative algorithm. The shadow vertex method is a simplex method to solve LP problems. It first projects the feasible set into a two-dimensional plane where the projection
of the optimal vertex is still a vertex of the projection of the feasible set, and then walks along
the vertices of the projected set to find the optimal one. The model used in [4,5] requires
the b-vector of the considered linear programs to be entry-wise positive and a specific non-
degeneracy assumption to hold. Since neither assumption holds in our case, we reprove the
two main theorems of [4,5] under relaxed assumptions that better fit our context.

Re-computing the optimum is not always necessary when player 2 generates a new action,
or in other words, a new constraint is added to the existing LP. We show that the necessary
and sufficient condition that the security strategy remains unchanged in the new game is that the
old security strategy satisfies the new constraint. We further analyze the probability that the
old optimum remains optimal in the new LP, and find that if columns of the payoff matrix are
independently and identically distributed, then the probability that the old optimum remains
optimal in the new LP increases with respect to the number of constraints. This is because as
the number of constraints increases, the feasible set is getting smaller, and it is less possible
for the new constraint to cut the optimal vertex off. In security problems, it means that if
the attackers explore the vulnerabilities of the system in a stochastic way, as the number of
revealed attacking methods increases, it is more likely that the security strategy does not need
to be updated.

If the old optimum is not optimal any more in the new LP, we do not need to start the
search all over again. Instead, we can test the feasibility of the previously visited shadow
vertices one by one, find out the feasible one with the best objective value, and use it as the
original shadow vertex to start the search. We call the algorithm the iterative shadow vertex
method. If the columns of the payoff matrix are independently and identically distributed,
the average computational complexity of the iterative shadow vertex method is a fraction of
that of the regular shadow vertex method. Otherwise, if cutting off the optimal vertex does
not necessarily mean all shadow vertices are cut off, the iterative shadow vertex method’s
average computational complexity conditioned on the situation that the security strategy
needs to be updated is strictly less than the regular shadow vertex method’s conditional
average computational complexity.

In our simulation sections, we first build a numerical example such that all of the
assumptions are satisfied, and the simulation results demonstrate our probability analysis
and computational complexity analysis. We then studied a practical urban security problem
which violates all of our assumptions. In this case, although the probability analysis and the
computational computational complexity analysis are invalid, the iterative algorithm is in
fact applicable, and the simulation results show that the iterative shadow vertex method cuts
more than half of the computation time of the regular shadow vertex method.

The rest of the paper is organized as follows. Section 2 states the problem, and Sect. 3 dis-
cusses the shadow vertex simplex method with the presence of the probability vector variable.
Section 4 presents the necessary and sufficient condition of unchanging security strategy and
the corresponding possibility, and Sect. 5 provides the iterative shadow vertex method and
the computational complexity analysis. The simulation results are given in Sects. 6 and 7,
followed by the conclusion in Sect. 8.

2 Matrix Games with Growing Action Set on One Side

Let \( \mathbb{R}^n \) denote an \( n \)-dimensional real space. For a finite set \( S \), \( |S| \) denotes its cardinality,
and \( \Delta(S) \) is the set of all possible probabilities over \( S \). Vectors \( \mathbf{1} \) and \( \mathbf{0} \) are column vectors
(whose dimension will be clear from context) with all their elements to be one and zero,
respectively. $I_n$ is an $n$-dimensional identity matrix and $e_i$ is the $i$th column of $I_n$. Let $u$ and $v$ be two $n$-dimensional vectors. The plane spanned by $u$ and $v$ is denoted by $span(u,v)$, and the minor arc between $u$ and $v$ is denoted by $arc(u,v)$.

A matrix game is specified by a triple $(S, Q, G)$, where $S$ and $Q$ are finite sets denoting player 1 and 2’s action sets, respectively. The matrix $G \in \mathbb{R}^{|S|\times|Q|}$ is the payoff matrix whose element $G_{s,q}$ is player 1’s payoff, or player 2’s penalty, if players 1 and 2 play $s \in S$ and $q \in Q$, respectively. In this matrix game, player 1 and player 2 are the maximizer and the minimizer, respectively. This paper considers mixed strategies as players’ strategy space. Player 1’s mixed strategy $\bar{x} \in \Delta(S)$ is a probability over player 1’s action set $S$. Player 2’s mixed strategy $\bar{y} \in \Delta(Q)$ is defined in the same way. Player 1’s expected payoff is $\gamma(\bar{x}, \bar{y}) = E_{\bar{x},\bar{y}}(G_{s,q}) = \bar{x}^T G \bar{y}$. Since both players use mixed strategies, there always exists a Nash equilibrium $(\bar{x}^*, \bar{y}^*)$ such that

$$\gamma(\bar{x}, \bar{y}^*) \leq \gamma(\bar{x}^*, \bar{y}^*) \leq \gamma(\bar{x}^*, \bar{y}), \quad \forall \bar{x} \in \Delta(S) \bar{y} \in \Delta(Q),$$

When a Nash equilibrium exists, the maxmin value meets the minmax value of the game, i.e., $\max_{\bar{x} \in \Delta(S)} \min_{\bar{y} \in \Delta(Q)} \gamma(\bar{x}, \bar{y}) = \min_{\bar{y} \in \Delta(Q)} \max_{\bar{x} \in \Delta(S)} \gamma(\bar{x}, \bar{y})$. In this case, we say the game has a value $V$, and $\bar{x}^*$, $\bar{y}^*$ are also called the security strategy of players 1 and 2, respectively. Player 1’s security strategy can be computed by solving the following linear program [2].

$$V = \max_{\bar{x} \in \mathbb{R}^n, \ell \in \mathbb{R}} \ell \quad \text{s.t.} \quad G^T \bar{x} \geq \ell \mathbf{1},$$

$$\mathbf{1}^T \bar{x} = 1,$$

$$\bar{x} \geq 0. \quad \text{(1)}$$

Player 2’s security strategy can be computed by constructing a similar linear program.

This paper considers a special case when player 2’s actions are revealed gradually, and the size of player 2’s action set is potentially large. Let us suppose that the current size of player 2’s action set is $m$, and the size of player 1’s action set is $n$ which is fixed. At some point, player 2 develops a new action $q_{\text{new}}$, which introduces a new column $g \in \mathbb{R}^n$ to the payoff matrix $G \in \mathbb{R}^{n \times m}$. The change of the payoff matrix may result in the change of player 1’s security strategy, and hence, player 1 may need to rerun the linear program with extended payoff matrix, as below, to get the new security strategy $\bar{x}_{\text{new}}^*$.

$$V_{\text{new}} = \max_{\bar{x} \in \mathbb{R}^n, \ell \in \mathbb{R}} \ell \quad \text{s.t.} \quad G^T \bar{x} \geq \ell \mathbf{1},$$

$$g^T \bar{x} \geq \ell \quad \text{(6)}$$

$$\mathbf{1}^T \bar{x} = 1 \quad \text{(7)}$$

Comparing the new LP (5–9) with the old one (1–4), we see that the only difference is the new LP adds one more constraint, Eq. (7), to the old LP. Our objective is to find an efficient way to compute the new security strategy of player 1.

Our approach consists of three steps. First, considering the potentially large size of player 2’s action set, in other words, a potentially large number of constraints in the LP problem, we propose to use the shadow vertex simplex method, whose average computational complexity is low with respect to the constraint size, as the basic method to solve the LP. Second, we
propose a necessary and sufficient condition guaranteeing player 1’s security strategy remains unchanged. If such a condition is violated, we then propose an iterative algorithm to reduce the computational time based on the transient results of the previous computation process.

3 Shadow Vertex Simplex Method with a Probability Variable

The shadow vertex simplex method was introduced in [4,5] and is motivated by the observation that the simplex method is very simple in two dimensions when the feasible set forms a polygon. In this case, simplex method visits the adjacent vertices, whose total number is usually small in two-dimensional cases, until it reaches an optimum vertex. The basic idea of the shadow vertex simplex method consists of two steps. It first projects a high-dimensional feasible set to a two-dimensional plane such that the projection of an optimum vertex is still a vertex of the projection (or shadow) of the feasible set. Then it walks along the adjacent vertices whose projections are also vertices of the shadow of the feasible set until it reaches an optimum vertex.

In order to use the shadow vertex method, we need to transform the LP (1–4) into a canonical form. The canonical form of LP (5–9) can be easily derived by replacing $G$ in (10–12) with $[G g]$. Let $x = [\bar{x}_1 \bar{x}_2 \ldots, \bar{x}_{n-1} \ell]^T$. We have $\bar{x} = [T \mathbf{0}]x + e_n$, where $T = [I_{n-1} - \mathbf{1}]^T$, and $e_n$ is the $n$th column of $I_n$. LP (1–4) is rewritten as follows.

$$V = \max_{x \in \mathbb{R}^n} c^T x$$

s.t. $Ax \leq b$, \hspace{1cm} (10)

where $c = [0^T 1]^T$, $b = [e_n^T G 1 0^T]^T$, and

$$A = \begin{bmatrix} -G^T T & 1 \\ -T & 0 \\ -I_{n-1} & 0 \end{bmatrix}. \hspace{1cm} (12)$$

We call the first $m$ constraints in (10, 11) the normal constraints and the last $n$ constraints the probability constraints. Notice that the last $n$ constraints cannot be active at the same time.

3.1 Initial Shadow Vertex, Projection Plane, and Initial Searching Table

Denote the feasible set of LP (10, 11) by $X$. We set $x_0 = [\mathbf{0} \min_{i = 1, \ldots, m} G_{n,i}]^T$ to be the initial vertex. Given a feasible solution $x \in \mathbb{R}^n$, we say constraint $i$ is active if $A_i x = b_i$. Let $\Omega_0 = \{i : A_i x_0 = b_i\}$ be the initial active constraint set, where $A_i$ is the $i$th row of $A$. Next, we will introduce the necessary and sufficient condition of optimality in terms of active constraint set. The lemma is similar to Lemma 1.1 of [5] (page 63) except that we don’t require $b$ to be positive. The proof is also similar and is given in Appendix for the completeness of this paper.

**Lemma 1** Consider a general linear program in canonical form.

$$\max_{x \in \mathbb{R}^n} c^T x$$

s.t. $Ax \leq b$, \hspace{1cm} (13)

where $A \in \mathbb{R}^{m \times n}$. Let $x^*$ be a vertex and $\Omega$ be the corresponding active constraint set. Then $x^*$ is an optimal solution with respect to $w^T x$ for some $w \in \mathbb{R}^n \setminus \{0\}$ if and only if there exists a nonnegative vector $\rho \in \mathbb{R}^{1|\Omega|}$ such that...
where $\Omega^i$ is the $i$th element in $\Omega$.

From Lemma 1, we see that searching for an optimal vertex is the same as looking for the active constraints whose convex cone contains the objective vector $c$. In contrast to Dantzig’s simplex method, which modifies basic variables iteratively until optimality conditions are satisfied, the shadow vertex method searches for active constraints. To make the searching process efficient, the shadow vertex method limits its search to shadow vertices which are defined as follows.

**Definition 1** Let $u, v$ be two linearly independent vectors and $\Gamma$ be the orthogonal projection onto $\text{span}(u, v)$. A vertex $x$ of a polygon $X$ is called a shadow vertex with respect to $\text{span}(u, v)$ if $\Gamma(x)$ is a vertex of $\Gamma(X)$.

An important issue is how to design the two-dimensional projection plane $\text{span}(u, v)$ such that both the initial vertex and the optimum vertex are shadow vertices. To this end, we introduce the relaxed non-degeneracy assumption and a necessary and sufficient condition for a vertex of the feasible set to be a shadow vertex.

**Assumption 1** For every vertex $x$ of the feasible set of the general linear program (13, 14) and the corresponding active constraint set $\Omega = \{i : A_i x = b_i\}$, there are only $n$ elements in $\Omega$, and any $n$ element subset of $\{A_{\Omega^1}, A_{\Omega^2}, \ldots, A_{\Omega^n}, u, c\}$ is linearly independent.

Such non-degeneracy assumptions are typical in the literature when rigorously proving complexity results, even though it is not believed that degenerate problems behave significantly worse [5]. While it is relatively hard to check whether Assumption 1 is satisfied, in a special case when $A_i$’s are distributed independently, identically and symmetrically under rotations, the probability of degeneracy is almost 0 [5]. The main difference between our relaxed non-degeneracy assumption and the non-degeneracy assumption in [5] is that we only require that for any vertex, any $n$ elements of $u, c$, and the $n$ active constraint rows of the vertex are linearly independent, while the non-degeneracy assumption in [5] requires any $n$ elements of $u, c$ and all constraint rows to be linearly independent. This difference doesn’t influence the necessary and sufficient condition of a shadow vertex (Lemma 1.2 on page 64 of [5]), and the proof is also similar. We give the proposition as follows, and the proof is given in Appendix for the completeness of this paper.

**Proposition 1** Consider the LP problem (10, 11), and suppose Assumption 1 holds. Let $\hat{x}$ be a vertex of the feasible set $X$ of (11), and $\Gamma : X \rightarrow \text{span}(u, v)$ be the orthogonal projection map from $X$ to $\text{span}(u, v)$. The following three conditions are equivalent.

1. $\hat{x}$ is a shadow vertex.
2. The projection of $\hat{x}$ is on the boundary of the projection of $X$, i.e., $\Gamma(\hat{x}) \in \partial \Gamma(X)$.
3. There exists a vector $w \in \mathbb{R}^n \setminus \{0\}$ in $\text{span}(u, v)$ such that $w^T \hat{x} = \max_{x' \in X} w^T x'$.

If Assumption 1 is violated, we still have results 2 and 3 based on result 1, but result 1 may not be true based on results 2 and 3. In other words, without Assumption 1, even if $\hat{x}$ is an optimum with respect to a vector in $\text{span}(u, v)$, $\hat{x}$ may not be a shadow vertex.

Proposition 1 implies that if we choose the projection plane to contain the objective vector $c$, then the optimum vertex is a shadow vertex. According to Lemma 1, if we construct an auxiliary objective vector $u = \sum_{i=1}^{\left|\Omega_0^i\right|} \rho_i A_{\Omega_0^i}^T \neq 0$ for some nonnegative $\rho_i$’s, which is linearly
independent of \( c \), then the initial vertex \( x_0 \) is optimal with respect to \( u \), and hence, \( x_0 \) is a shadow vertex with respect to \( \text{span}(u, c) \) according to the shadow vertex condition (1).

Now that we have found the initial vertex, the initial active constraint set and the projection plane, it is time to build the searching table. The basic idea of constructing the table is to find a linear combination of active constraint vectors for objective vector \( c \), auxiliary objective vector \( u \) and all other constraint vectors. Given any active constraint set \( \Omega_0 \), we construct the searching table in the following way. The first row consists of an \( n \)-dimensional row vector \( \alpha \) and a scalar \( Q_c \) which satisfies \( c = \sum_{j=1}^{n} \alpha_j A_{\Omega_0}^j \) and \( Q_c = -\sum_{j=1}^{n} \alpha_j b_{\Omega_0}^j \).

The second row consists of an \( n \)-dimensional row vector \( \beta \) and a scalar \( Q_u \) which satisfies \( u = \sum_{j=1}^{n} \beta_j A_{\Omega_0}^j \) and \( Q_u = -\sum_{j=1}^{n} \beta_j b_{\Omega_0}^j \). Each row for the next \( m + n \) rows consists of an \( n \)-dimensional row vector \( \gamma_i \) and a scalar \( \phi_i \) which satisfies \( A_i = \sum_{j=1}^{n} \gamma_{ij} A_{\Omega_0}^j \) and \( \phi_i = b_i - \sum_{j=1}^{n} \gamma_{ij} b_{\Omega_0}^j \). Notice that the objective value is \(-Q_c\) and the feasibility is implied by non-negativity of \( \phi_j \)'s. We build the initial table using the following algorithm.

**Algorithm 1 (Initialization)**

1. Find \( l \) such that \( G_{n,l} = \min_j G_{n,j} \).
2. Let \( \Omega_0 = \{l, m + 2, \ldots, m + n\} \).
3. Find \( \beta > 0 \) such that \( u = \sum_{j=1}^{n} \beta_j A_{\Omega_0}^j \) satisfies Assumption 1.
4. Compute \( Q_u = -\sum_{j=1}^{n} \beta_j b_{\Omega_0}^j \).
5. Compute \( \alpha \) such that \( c = \sum_{j=1}^{n} \alpha_j A_{\Omega_0}^j \).
   
   \[
   \alpha_j = \begin{cases} 
   1, & \text{if } j = 1 \\
   A_{l,j-1}, & \text{if } j \neq 1 
   \end{cases}
   \]
6. Compute \( Q_c = -\sum_{j=1}^{n} \alpha_j b_{\Omega_0}^j \).
7. Compute \( \gamma_i \) such that \( A_i = \sum_{j=1}^{n} \gamma_{ij} A_{\Omega_0}^j \), \( \forall i = 1, \ldots, m + n \).
   
   \[
   \gamma_{ij} = \begin{cases} 
   1, & \text{if } j = 1 \text{ and } i \leq m \\
   A_{l,j-1} - A_{i,j-1}, & \text{if } j \neq 1 \text{ and } i \leq m 
   \end{cases}
   \]

   \[
   \gamma_{m+1} = [0 \ldots -1],
   \gamma_{m+j} = e_j, \forall j = 2, \ldots, n.
   \]
8. Compute \( \phi_i = b_i - \sum_{j=1}^{n} \gamma_{ij} b_{\Omega_0}^j \), for all \( i = 1, \ldots, m + n \).
9. Record the initial table \( \text{table(0)} \) and the initial active constraint set \( \Omega_0 \).

### 3.2 Pivot Step

Suppose the current shadow vertex is \( x_t \), and the corresponding table and active constraint set \( A_t \) are given. We first check whether \( x_t \) is an optimum vertex. If not, we search for the adjacent shadow vertex with a larger objective value (if exists). This is called ‘taking a pivot step.’

According to Lemma 1, if \( \alpha_i \geq 0 \) for all \( i = 1, \ldots, n \), then \( x_t \) is an optimum vertex, and the search ends. Otherwise, we need to find an adjacent shadow vertex with a larger objective value. To this end, the following lemma is useful.

**Lemma 2** (Lemma 1.4 in [5] (page 68)) Consider the LP problem (10, 11). Suppose \( u \) is linearly independent with \( c \). Let \( x_0, x_1, \ldots, x_t \) be the optimal vertices with respect to
\[ w_i^T x_i, w_1^T x_1, \ldots, w_t^T x_t, \text{ where } w_i \in \text{span}(u, c) \setminus \{0\}, w_0 = u, \text{ and } \text{arc}(w_i, c) > \text{arc}(w_{i+1}, c) \text{ for } i = 0, \ldots, t - 1. \]

Then

\[ c_i^T x_i < c_{i+1}^T x_{i+1}, \quad \forall i = 0, \ldots, t - 1. \]

With Lemma 2, we build a vector \( w(\mu_t) = u + \mu c \). Let \( w_t = w(\mu_t) \) where \( \mu_t \) is the smallest nonnegative \( \mu \) such that the current shadow vertex \( x_t \) is a maximum with respect to \( w(\mu_t) x \). If \( t = 0 \), then \( \mu_t = 0 \). The basic idea of pivot is to increase \( \mu \) from \( \mu_t \) to \( \mu_{t+1} \) such that \( x_t \) won’t be a maximum with respect to \( w(\mu_{t+1}) + \varepsilon \) for any \( \varepsilon > 0 \) and then find the next shadow vertex \( x_{t+1} \) which is the maximal with respect to \( w_{t+1} = w(\mu_{t+1}) \). Since \( \text{arc}(w_t, c) > \text{arc}(w_{t+1}, c) \), we assure that \( x_{t+1} \) has a larger objective value than \( x_t \) according to Lemma 2. Based on this idea, the moving out active constraint \( \Omega_i^k \) is decided as follows.

\[
k = \arg \min_{i \in \{1, \ldots, n\}, \alpha_i < 0} \frac{\beta_i}{\alpha_i} \tag{16}\]

Interested readers can find more details in Lemma 1.11 and its proof on page 93 and 94 in [5].

Next, we will discuss which inactive constraint should be moved into the active constraint set to guarantee the feasibility. First of all, the moving-in constraint \( i \) should satisfy \( \gamma_{ik} < 0 \). If \( \gamma_{ik} \geq 0 \) for all \( i \), then \( A_k \) and all \( A_j \)’s for \( j \notin \Omega \) lie in the same half space divided by the hyperplane through points \( 0, A_{\Omega_i^k}, \ldots, A_{\Omega_i^{k-1}}, A_{\Omega_i^{k+1}}, \ldots, A_{\Omega_i^{m+n}} \), and \( c \) lies on the other half space. Therefore, \( c \) doesn’t lie in the convex cone of \( A_1, \ldots, A_{m+n} \), which means that the LP problem has no solution, and the search stops. If there exists at least one inactive constraint \( i \) such that \( \gamma_{ik} < 0 \), let us suppose that we choose constraint \( l \) satisfying \( \gamma_{ik} < 0 \) as the moving-in constraint. The table will be updated as follows [5].

\[
\alpha_j = \begin{cases} 
\alpha_j - \alpha_k \frac{\gamma_{lj}}{\gamma_{lk}} & \text{if } j \neq k \\
\alpha_k & \text{if } j = k
\end{cases} \tag{17}
\]

\[
\beta_j = \begin{cases} 
\beta_j - \frac{\beta_k}{\gamma_{lk}} \frac{\gamma_{lj}}{\gamma_{lk}} & \text{if } j \neq k \\
\frac{\beta_k}{\gamma_{lk}} & \text{if } j = k
\end{cases} \tag{18}
\]

\[
\gamma_{ij} = \begin{cases} 
\gamma_{ij} - \gamma_{ik} \frac{\gamma_{lj}}{\gamma_{lk}} & \text{if } j \neq k \text{ and } i \notin \Omega_{l+1} \\
\frac{\gamma_{lj}}{\gamma_{lk}} & \text{if } j = k \text{ and } i \notin \Omega_{l+1}
\end{cases} \tag{19}
\]

\[
\gamma_i = e_k, \text{ if } i = l. \tag{20}
\]

and

\[
\phi_i = \begin{cases} 
0 & \text{if } i \in \Omega_{l+1} \\
\phi_i - \gamma_{ik} \phi_l & \text{if } i \notin \Omega_{l+1}
\end{cases} \tag{21}
\]

\[
Q_c = Q_c - \frac{\phi_l \alpha_k}{\gamma_{lk}} \tag{22}
\]

\[
Q_u = Q_u - \frac{\phi_l \beta_k}{\gamma_{lk}} \tag{23}
\]

As mentioned before, the non-negativity of \( \phi_i \)’s implies the feasibility. Therefore, according to (21) and the analysis above, the moving-in constraint \( l \) is chosen such that

\[
l = \arg \max_{i \notin \Omega, \gamma_{ik} < 0} \frac{\phi_l}{\gamma_{ik}} \tag{24}
\]
and the active constraint set is updated to
\[ \Omega_{t+1} = \{ \Omega_t^1, \ldots, \Omega_t^{k-1}, l, \Omega_t^{k+1}, \ldots, \Omega_t^n \}. \]

The pivot algorithm given the current active constraint set \( \Omega_t \) is provided as follows.

**Algorithm 2 (Pivot)**

1. If \( \alpha_i \geq 0 \) for all \( i \), then the vertex associated with \( \Omega_t \) is the optimum, and \( v = -Q_c. \) Go to step 8).
2. Find the moving out constraint \( \Omega_k^t \), where \( k \) is given in (16).
3. If \( \gamma_{ik} \geq 0 \) for all \( i \neq \Omega_t \), then there is no solution. Go to step 8).
4. Find the moving-in constraint \( l \) satisfying (24).
5. Update and record \( \Omega_{t+1} = \{ \Omega_1^t, \ldots, \Omega_{k-1}^t, l, \Omega_{k+1}^t, \ldots, \Omega_n^t \} \).
6. Update the table according to (17–23), and record it as table \((t + 1)\).
7. Return to step 1).
8. End.

The shadow vertex simplex method has a polynomial average computational complexity [5]. To be more specific, let \( \tau \) be the number of pivot steps and \( T \) be the number of shadow vertices. If \( A_1, A_2, \ldots, A_m \) and \( c \) are independently, identically, and symmetrically distributed under rotation, then \( E(\tau) \leq E(T) \leq \rho m^{1-1/n} n^3 \) for some positive constant \( \rho \).

**4 Player 1’s Unchanging Security Strategy**

It is not necessary to run the shadow vertex simplex method every time player 2 adds a new action since player 2’s new action may have no influence on player 1’s security strategy and the game value. Comparing the old LP (1–4) with the new LP (5–9), we see that the new LP adds a new constraint (7) to the old LP. Geometrically, a new constraint means a new cut of the existing feasibility set. If the new constraint does not cut the optimal vertex off, i.e., the optimal vertex satisfies the new constraint, then the optimal vertex remains the same, and player 1’s security strategy does not change. To formally state the above analysis and analyze the probability of a changing security strategy, we first provide the canonical form of the new LP (5–9), then list the assumptions to analyze the probability, and finally give the main results of this section in Theorem 1.

As mentioned before, the canonical form of (5–9) takes the same form as in (10, 11) except that the payoff matrix is replaced with \([G g]\). To make the discussion clear, we provide the canonical form of (5–9) as follows.

\[
V_{\text{new}} = \max_{x \in \mathbb{R}^n} c^T x \tag{25}
\]
\[
\text{s.t. } \tilde{A}x \leq \tilde{b} \tag{26}
\]

where
\[
\tilde{A} = \left[ A_1^T \left[ g_n - g_1 \ g_n - g_2 \ \cdots \ g_n - g_{n-1} \ 1 \right]^T A_{m+1:m+n}^T \right]^T,
\]
and
\[
\tilde{b} = \left[ b_1^T \ g_n \ b_{m+1:m+n}^T \right]^T.
\]

The new constraint in the new LP (5–9) is transformed to
\[
[g_n - g_1 \ g_n - g_2 \ \cdots \ g_n - g_{n-1} \ 1]x \leq g_n. \tag{27}
\]
To analyze the probability of a changing security strategy, we make the following assumptions.

**Assumption 2** The columns of the extended payoff matrix \([G, g]\) are independently and identically distributed.

This kind of assumption is common in the literature on linear programming, particularly in works concerned with smooth/average computational complexity analysis [5,12,13,15,16]. In the context of security games that interests us, Assumption 2 is satisfied in the presence of an 'unsophisticated' attacker, who explores the vulnerabilities of the system in a stochastic way. When vulnerabilities, and hence the new columns of the payoff matrix, are not determined in an i.i.d. fashion, the algorithm we propose is still applicable, but some of the theoretical complexity guarantees may not hold, even though they still appeared to be verified empirically.

**Assumption 3** The old LP problem \((10, 11)\) has a unique optimal solution.

To the best of our knowledge, there are no results regarding the probability of a unique solution in LP problems. To test how strict this assumption is, we use several different probability distributions to generate the payoff matrix and use simplex method to solve the LP problem \((10, 11)\) to see how often the assumption is violated (Please check Chapter 5-1 of [6] for the details about how to test uniqueness of a solution using the simplex method). In our test, for every probability distribution, all the columns of a \(2 \times 10\) payoff matrix are generated independently and identically. For uniformly distributed integer vectors ranging from \([-100, -100]^T\) to \([100, 100]^T\), the possibility of unique optimal solution is 99%. For uniformly distributed real vectors ranging from \([-100, -100]^T\) to \([100, 100]^T\), the possibility of unique optimal solution is almost 1 (all 5000 randomly generated payoff matrices have unique solutions). For normally distributed two-dimensional vectors with mean \([0, 0]^T\) and covariance \(I_2\), the possibility of unique optimal solution is almost 1 (all 5000 randomly generated payoff matrices have unique solutions). The possibility remains the same if we vary the mean value and the covariance matrix. From these empirical tests, Assumption 3 thus does not appear to be overly restrictive for families of randomly generated LP matrices. Note that, just like Assumptions 2 and 3 is only needed to provide theoretical computational complexity guarantees, but in no way restricts the applicability of our method.

Notice that while Assumption 1 is more general than the non-degenerate assumption made in [4,5], Assumption 2 and 3 is neither more nor less general than the probability assumption in [4,5], which assumes that \(c, \hat{A}_1, \ldots, \hat{A}_{m+n}\) are distributed identically, independently, and symmetrically under rotation. Since in our model, \(c\) and the probability constraints \(\hat{A}_{m+1}, \ldots, \hat{A}_{m+n}\) are fixed, our model does not satisfy the probability assumption in [4,5]. Meanwhile, a model satisfies the probability assumption in [4,5] does not necessarily satisfy Assumption 3, the unique optimal solution assumption.

We are now in a position to state necessary and sufficient conditions for the modification of the LP's optimal solution. Under Assumptions 1–3, we can further estimate the probability of the optimal vertex changing.

**Theorem 1** Let \(x^*\) be the optimal solution of old LP \((10, 11)\). The old optimal solution \(x^*\) remains optimal in the new LP \((25, 26)\) if and only if \(x^*\) satisfies the new constraint \((27)\).

Moreover, if Assumptions 1–3 hold, the probability that player 1’s security strategy changes satisfies

\[
P(\text{player 1’s security strategy changes}) \leq \frac{n}{m + 1}. \tag{28}\]
Proof It is easy to see that if $x^*$ is an optimal solution of (25, 26), then $x^*$ satisfies the new constraint.

For the opposite direction, suppose $x^*$ satisfies the new constraint. With one more constraint, the feasible set of the new LP (25, 26) is included in the feasible set of the old LP (10, 11). So we have $V_{\text{new}} \leq V$, in other words, $cx^+ \leq cx^*$, where $x^+$ is an optimal solution of the new LP (25, 26).

Meanwhile, since $[g_n - g_1, g_n - g_2, \ldots, g_n - g_{n-1}]x^* \leq g_n$, it is easy to see that $x^*$ is a feasible solution of the new LP (25, 26), and we have $cx^* \leq cx^+$.

Therefore, $cx^* = cx^+$, and $x^*$ is an optimal solution of the new (25, 26).

By now, we have shown the necessary and sufficient condition of an unchanging security strategy is that $x^*$ satisfies the new constraint. Next, we will show that the probability of changing the security strategy is bounded from above by $\frac{n}{m+1}$.

First of all, based on Assumption 2 and Lemma 5, we have

$$P(\Omega_1 \text{ is the optimal active constraint set}) = P(\Omega_2 \text{ is the optimal active constraint set}) = \varepsilon_{\kappa_{\ell}, \ldots, \kappa_{n-\ell}},$$

where $\Omega_1$ and $\Omega_2$ are defined as in (40) and (41), respectively.

Next, we notice that

$$\sum_{\Omega \in [1, \ldots, m+1+n]} P(\Omega \text{ is the optimal active constraint set}) = 1.$$

To be more specific, we have

$$1 = \sum_{\Omega \in [1, \ldots, m+1+n], \Omega \cap [m+2, \ldots, m+1+n] = \emptyset} P(\Omega \text{ is the optimal active constraint set})$$

$$+ \sum_{\Omega \in [1, \ldots, m+1+n], \Omega \cap [m+2, \ldots, m+1+n] = m+2} P(\Omega \text{ is the optimal active constraint set})$$

$$+ \cdots$$

$$+ \sum_{\Omega \in [1, \ldots, m+1+n], \Omega \cap [m+2, \ldots, m+1+n] = [m+3, \ldots, m+1+n]} P(\Omega \text{ is the optimal active constraint set})$$

$$= \left(\frac{m+1}{n}\right)\varepsilon_\emptyset + \left(\frac{m+1}{n-1}\right)\varepsilon_{m+2} + \cdots + \left(\frac{m+1}{1}\right)\varepsilon_{m+3, \ldots, m+1+n}.$$

(29)

According to Lemma 4, the event that $x^*$ is not optimal in the new LP problem is the same as the event that the new constraint is active for any optimum of the new LP. Therefore, we have

$$P(x^* \text{ is not optimal}) = P(\text{constraint } m + 1 \text{ is in the optimal active constraint set}).$$
Taking a further look at this probability, we have

\[
P(\text{constraint } m + 1 \text{ is in the optimal active constraint set})
\]

\[
= \sum_{\Omega \in \{1, \ldots, m+1+n\}} P(\Omega \text{ is the optimal active constraint set})
\]

\[
+ \sum_{\Omega \in \{1, \ldots, m+1+n\}, \Omega \cap [m+2, \ldots, m+1+n] = \emptyset} P(\Omega \text{ is the optimal active constraint set})
\]

\[
+ \cdots
\]

\[
+ \sum_{\Omega \in \{1, \ldots, m+1+n\}, \Omega \cap [m+2, \ldots, m+1+n] = m+2} P(\Omega \text{ is the optimal active constraint set})
\]

\[
\frac{m}{n - 1} \varepsilon_{m+1} \leq \frac{n}{m + 1},
\]

Comparing Eq. (29) with Eq. (30), with a simple mathematical derivation, we have

\[
P(\text{constraint } m + 1 \text{ is in the optimal active constraint set}) \leq \frac{n}{m + 1},
\]

which completes the proof.

From the probability analysis in Theorem 1, we noticed that as \(m\) grows, the probability to re-compute the security strategy decreases. This is because as the size of constraints grows, the feasible set shrinks, and it is less possible for the new constraint to cut the optimal vertex off. In security problems, it means that if the attackers explore the vulnerabilities of the system in a stochastic way, as the number of the revealed attacking actions increases, it is more likely that there is no need to update the security strategy.

5 Iterative Shadow Vertex Simplex Method

If the old security strategy is not optimal any more in the new game, it is not always necessary to start the optimum search from the initial vertex. Suppose that we have constructed a complete shadow vertex path \(\Pi = \{x_0, x_1, \ldots, x_{\tau-1}, x^*\}\) when searching for the optimum of the old LP (10, 11). When the new constraint (27) is added, we can test the feasibility of the visited shadow vertices one by one starting from \(x_0\) until we find one that fails the feasibility test. The last feasible vertex should have the largest objective value, and will be chosen as the starting point of the new search. This idea is formulated as the iterative shadow vertex simplex method as follows.

**Algorithm 3** (Iterative shadow vertex algorithm)

1. If \(x^*\) satisfies (27), then \(x^*\) is the optimal solution. Go to step 6).
2. Otherwise, set the new path \(\Pi_{\text{new}} = \emptyset\).
3. From \(i = 0\) to \(\tau - 1\), test whether \(x_i\) satisfies the new constraint (27). If yes, add \(x_i\) into the new path \(\Pi_{\text{new}}\), and update table \(i\) by inserting a new row vector \(\gamma_{m+1}\) and a new scalar \(\phi_{m+1}\) with respect to the new constraint. Otherwise, stop the test.
4. If \( \Pi_{\text{new}} = \emptyset \), then run standard shadow vertex algorithm, i.e., run the initialization algorithm 1, and then the pivot algorithm 2. Add all visited shadow vertices including the optimal vertex into \( \Pi_{\text{new}} \). Update \( \Pi = \Pi_{\text{new}} \). Go to step 6.

5. Otherwise, choose the last element in \( \Pi_{\text{new}} \) as the initial shadow vertex, and run the pivot algorithm 2. Add all visited shadow vertices including the optimal vertex to \( \Pi_{\text{new}} \). Update \( \Pi = \Pi_{\text{new}} \). Go to step 6.

6. End.

Since the iterative shadow vertex simplex method usually starts the search from a shadow vertex with a larger objective value, we expect that the iterative shadow vertex simplex method has less pivot steps and lower computational complexity than the shadow vertex simplex method. To analyze the computational complexity of the former, we start from the following lemma stating that if the new constraint does not cut the whole path off, then with the iterative shadow vertex method, the new constraint stays in the active constraint set until the optimum vertex is found.

**Lemma 3** Suppose Assumption 1 holds, and the optimum \( x^* \) of the old LP (10, 11) violates the new constraint (27). Let \( \Pi = \{ x_0, x_1, \ldots, x_{\tau - 1}, x^* \} \) be a complete search path of the old LP (10, 11), i.e., \( \{ x_0, x_1, \ldots, x_{\tau - 1}, x^* \} \) is a sequence of adjacent shadow vertices, and \( \Omega_0, \ldots, \Omega_\tau \) be the corresponding active constraint set. Denote \( x_t \in \Pi \) the last shadow vertex in path \( \Pi \) satisfying the new constraint, i.e.,

\[
\begin{align*}
    c^T x_t &= \max_{x_i \in \Pi} c^T x_i \\
    \text{s.t. } &\hat{A}_{m+1} x_t \leq \hat{b}_{m+1}. 
\end{align*}
\]

With the iterative shadow vertex simplex Algorithm 3, let \( \Pi_{\text{new}} = \{ x_0, x_1, \ldots, x_t, \hat{x}_{t+1}, \ldots, \hat{x}_{\tilde{t}-1}, x^* \} \) be the search path of the new LP (25, 26), and \( \Omega_0, \Omega_1, \ldots, \Omega_t, \Omega_{\tilde{t}+1}, \ldots, \Omega_{\tau} \) be the corresponding active constraint sets. The new constraint (27) stays in the active constraint sets \( \Omega_{t+1}, \ldots, \Omega_{\tilde{t}}, \) i.e., \( m+1 \in \Omega_{i} \) for all \( i = t+1, \ldots, \tilde{t} \).

**Proof** First, we show that \( \hat{\Omega}_{t+1} \) contains \( m+1 \). Suppose in the search path \( \Pi \) from step \( x_t \) to \( x_{t+1} \), the move-out constraint is \( k \) and the move-in constraint is \( l \). In the search process of the new LP from \( x_t \) to \( \hat{x}_{t+1} \), if the new constraint \( \hat{A}_{m+1} x \leq \hat{b}_{m+1} \) is not chosen either because \( \gamma_{m+1,k} \geq 0 \) or because \( \phi_{m+1,k}/\gamma_{m+1,k} \) is not the maximal one, \( l \) will be chosen again to enter the active constraint set which implies that \( \hat{x}_{t+1} = x_{t+1} \) based on Assumption 1. Since the pivot step guarantees feasibility, we have \( x_{t+1} \) satisfies the new constraint. Meanwhile, we also have \( c^T x_{t+1} > c^T x_t \), which contradicts Eqs. (31, 32). Therefore, \( m+1 \) must be in \( \hat{\Omega}_{t+1} \).

Next, we show that once constraint \( m+1 \) enters the active constraint set, it will stay there until the optimum vertex is found. Let us suppose that \( \hat{\Omega}_{t+i} \) contains \( m+1 \), but \( \hat{\Omega}_{t+i+1} \) does not, for some \( i = 1, \ldots, \tilde{t} - t - 1 \). Since \( x_t \) and \( \hat{x}_{t+i+1} \) are both shadow vertices, there must exist \( w_t, w_{t+i+1} \in \text{span}(u, c) \) such that \( x_t, \hat{x}_{t+i+1} \) are the maximum with respect to \( w_t \) and \( w_{t+i+1} \), respectively, according to Proposition 1. The pivot rule of shadow vertex simplex method guarantees that \( \text{arc}(w_t, c) > \text{arc}(w_{t+i+1}, c) \) and \( w_{t+i+1} \) is in the convex cone formed by \( w_t \) and \( c \). Meanwhile, \( \hat{x}_{t+i+1} \) guarantees value improvement and feasibility, i.e., \( c^T \hat{x}_{t+i+1} > c^T x_t \) and \( \hat{A}_{m+1} \hat{x}_{t+i+1} \leq \hat{b}_{m+1} \). According to (31, 32), we see that \( \hat{x}_{t+i+1} \) is not in the search path \( \Pi \) of the old LP, and \( w_{t+i+1} \) does not lie in the convex cone formed by \( w_t \) and \( c \), which contradicts the previous conclusion. Therefore, if \( \hat{\Omega}_{t+i} \) contains \( m+1 \), so does \( \hat{\Omega}_{t+i+1} \) for all \( i = 1, \ldots, \tilde{t} - t - 1 \). \( \Box \)

Lemma 3 implies that the iterative shadow vertex simplex method only visits shadow vertices whose active constraint set contains \( m+1 \), and the number of pivot steps \( \tilde{t} \) can
thus be no greater than the number of shadow vertices $\hat{T}$ whose active constraint set contains $m + 1$. Based on the same assumptions and the similar idea as those used in the probability analysis in Theorem 1, we can further analyze the expected value of $\hat{T}$.

**Theorem 2**  
Consider the new LP problem (25, 26). Suppose Assumptions 1–3 hold. We have

$$E(\hat{T}) \leq \frac{n}{m + 1} E(T).$$  

(33)

If only Assumption 1 holds, and if the event that the new constraint cuts off the optimum $x^*$ of the old LP (10, 11) does not imply that the shadow vertices of the old LP (10, 11) are all cut off, i.e.,

$$P(\hat{A}_{m+1} x_1 > \hat{b}_{m+1}, \ldots, \hat{A}_{m+1} x_T > \hat{b}_{m+1} | \hat{A}_{m+1} x^* > \hat{b}_{m+1}) < 1,$$

(34)

where $x_i$ for $i = 1, \ldots, T$ denotes the shadow vertices of the old LP (10, 11), then we have

$$E(\hat{T} | \hat{A}_{m+1} x^* > \hat{b}_{m+1}) < E(T | \hat{A}_{m+1} x^* > \hat{b}_{m+1}).$$

(35)

**Proof**  
We first prove Eq. (33). Let $\Omega$ be an $n$ element subset in $\{1, \ldots, n\}$. We have

$$E(T) = \sum_{\Omega \in \{1, \ldots, m+1+n\}} P(\hat{A}_\Omega \text{ forms a shadow vertex}).$$

According to Lemma 6, we have

$$P(\Omega_1 \text{ forms a shadow vertex}) = P(\Omega_2 \text{ forms a shadow vertex}) \doteq \xi_{\kappa_1, \ldots, \kappa_{n-\ell}},$$

where $\Omega_1$ and $\Omega_2$ are defined as in (40) and (41), respectively.

Take a further look at $E(T)$, we have

$$E(T) = \sum_{\Omega \in \{1, \ldots, m+1+n\}, \Omega \cap \{m+2, \ldots, m+1+n\} = \emptyset} P(\Omega \text{ forms a shadow vertex})$$

$$+ \sum_{\Omega \in \{1, \ldots, m+1+n\}, \Omega \cap \{m+2, \ldots, m+1+n\} = \{m+2\}} P(\Omega \text{ forms a shadow vertex})$$

$$+ \cdots$$

$$+ \sum_{\Omega \in \{1, \ldots, m+1+n\}, \Omega \cap \{m+2, \ldots, m+1+n\} = \{m+3, \ldots, m+1+n\}} P(\Omega \text{ forms a shadow vertex})$$

$$= \left( m + 1 \right) \xi_0 + \left( m + 1 \right) \xi_{m+2} + \cdots + \left( m + 1 \right) \xi_{m+3, \ldots, m+1+n}. \quad (36)$$
Similar to the analysis of $E(T)$, for the expected number of shadow vertices whose active constraint set contains $m + 1$, we have
\[
E(\hat{T}) = \sum_{\Omega \in \{1, \ldots, m+1+n\}, \Omega \cap [m+2, \ldots, m+1+n] = \emptyset, m+1 \in \Omega} P(\Omega \text{ forms a shadow vertex})
+ \sum_{\Omega \in \{1, \ldots, m+1+n\}, \Omega \cap [m+2, \ldots, m+1+n] = [m+2], m+1 \in \Omega} P(\Omega \text{ forms a shadow vertex})
+ \cdots
+ \sum_{\Omega \in \{1, \ldots, m+1+n\}, \Omega \cap [m+2, \ldots, m+1+n] = [m+3, \ldots, m+1+n], m+1 \in \Omega} P(\Omega \text{ forms a shadow vertex})
= \left( \frac{m}{n-1} \right) \xi_0 + \left( \frac{m}{n-2} \right) \xi_2 + \cdots + \left( \frac{m}{0} \right) \xi_m
\tag{37}
\]
Comparing Eqs. (37) with (36), we see that $E(\hat{T}) \leq \frac{n}{m+1} E(T)$.
Next, we prove Eq. (35).
\[
E(\hat{T} | \hat{A}_{m+1}x^* > \hat{b}_{m+1}) = \sum_{\Omega \in \{1, \ldots, m+1+n\}, m+1 \in \Omega} P(\Omega \text{ forms a shadow vertex} | \hat{A}_{m+1}x^* > \hat{b}_{m+1}).
\]
Similarly, we have
\[
E(T | \hat{A}_{m+1}x^* > \hat{b}_{m+1})
= \sum_{\Omega \in \{1, \ldots, m+1+n\}, m+1 \in \Omega} P(\Omega \text{ forms a shadow vertex} | \hat{A}_{m+1}x^* > \hat{b}_{m+1})
+ \sum_{\Omega \in \{1, \ldots, m+1+n\}, n+1 \notin \Omega} P(\Omega \text{ forms a shadow vertex} | \hat{A}_{m+1}x^* > \hat{b}_{m+1})
= E(\hat{T} | \hat{A}_{m+1}x^* > \hat{b}_{m+1}) + \sum_{\Omega \in \{1, \ldots, m+1+n\}, m+1 \notin \Omega} P(\Omega \text{ forms a shadow vertex} | \hat{A}_{m+1}x^* > \hat{b}_{m+1})
\]
Generally speaking, the second term is nonnegative, and hence, we have $E(T | \hat{A}_{m+1}x^* > \hat{b}_{m+1}) \geq E(T | \hat{A}_{m+1}x^* > \hat{b}_{m+1})$. If cutting off the old optimum does not necessarily means all shadow vertices are cut, i.e., Eq. (34) holds, then it is always possible to have at least one shadow vertex in the old LP problem which is not cut off by the new constraint (27), i.e.,
\[
P(\exists x_i, \text{ s.t. } \hat{A}_{m+1}x^* \leq \hat{b}_{m+1} | \hat{A}_{m+1}x^* > \hat{b}_{m+1}) > 0.
\]
Since the shadow vertex of the old LP problem does not contain the new constraint $m + 1$, we have $\sum_{\Omega \in \{1, \ldots, m+1+n\}, n+1 \notin \Omega} P(\Omega \text{ forms a shadow vertex} | \hat{A}_{m+1}x^* > \hat{b}_{m+1}) > 0$, and hence Eq. (35) is true.$\square$

Theorem 2 includes two results, the average computational complexity analysis $E(\hat{T})$, and the conditional computational complexity analysis $E(\hat{T} | \hat{A}_{m+1}x^* > \hat{b}_{m+1})$. Both results are based on Assumption 1, a necessary condition to make sure both the regular shadow vertex
method and the iterative shadow vertex method only visit shadow vertices to find the optimal vertex. If Assumptions 2 and 3 are also satisfied, the average computational complexity of the iterative shadow vertex method is a fraction of the average computational complexity of the regular shadow vertex method. Otherwise, if cutting of the optimal vertex does not necessarily mean all the shadow vertices are cut off, the conditional average computational complexity of the iterative shadow vertex method is strictly less than the conditional average computational complexity of the regular shadow vertex method.

Notice that the average computational complexity of the shadow vertex simplex method is $O\left(m^{1/3}n^{1/3}\right)$ as shown in [5], while that of the iterative shadow vertex method is $O\left(m^{1/3}n^{4/3}\right)$. If $n > 2$, then the computational complexity of the iterative shadow vertex decreases as $m$ grows. This observation coincides with the observation based on the probability analysis in Theorem 1 which shows that as $m$ grows, it is more possible for the old optimum to be optimal in the new LP problem. The higher possibility that the old optimum remains optimal in the new LP problem leads to higher possibility of zero pivot step, and hence, the average number of the pivot steps is decreased. In security problems, it means that if the attackers explore the vulnerabilities of the systems in a stochastic way, as the number of the revealed attacking methods increases, the average computation time of the security strategy is reduced. It is mainly because the possibility to update the security strategy is decreasing as the number of the revealed attacking methods increases.

The above analysis assumes that we have a complete search path $\Pi$ of the old LP. Sometimes, the new constraint may cut off a part of the search path, but not the optimal vertex. In this case, we can immediately provide the optimal vertex and then repair the search path in the background to get prepared for player 2’s next new action. The idea of repairing the search path is similar to the iterative shadow vertex method, and we provide the algorithm as follows.

**Algorithm 4** *(Search path repair algorithm)*

1. If $x^*$ violates the new constraint $eq_{\text{new constraint}}$, go to step 8).
2. Set $\Pi_{\text{new}} = \emptyset$ and $\text{table}_{\text{new}} = \emptyset$.
3. From $i = 0$ to $\tau - 1$, if $x_j$ satisfies the new constraint, then add $x_j$ into the new path $\Pi_{\text{new}}$ and update $\text{table}_{\text{new}}(i)$ by inserting a new row vector $\gamma_{m+1}$ and a new scalar $\phi_{m+1}$ with respect to the new constraint to $\text{table}(i)$. Otherwise, stop the test.
4. If all the elements in $\Pi$ satisfy the new constraint, go to step 8).
5. Repair the search path.
   1. Let $j = -1$.
   2. Let $j = j + 1$. Choose the last element in $\Pi_{\text{new}}$ as the initial shadow vertex, and run step 2)–7) of the pivot algorithm, Algorithm 2. Add the visited shadow vertex $x_{i+j}$ to $\Pi_{\text{new}}$ and update $\text{table}_{\text{new}}(i + j)$.
   3. If $x_{i+j} \in \Pi \text{ or } x_{i+j}$ is the optimal vertex, then go to step 6). Otherwise, go to step b).
6. Add all the visited shadow vertices after $x_{i+j}$ in path $\Pi$ to the new path $\Pi_{\text{new}}$. Update the corresponding tables by inserting a new row vector $\gamma_{m+1}$ and a new scalar $\phi_{m+1}$ with respect to the new constraint, and add the updated tables to $\text{table}_{\text{new}}$.
7. Let $\text{table} = \text{table}_{\text{new}}$, and $\Pi = \Pi_{\text{new}}$.
8. End.
6 Numerical Examples

In this section, we consider several numerical examples to demonstrate the computational properties of the iterative shadow vertex method and compare them to our theoretical predictions in Sects. 4 and 5.

We first generate a $10 \times 100$ random payoff matrix whose elements are identically, independently, and uniformly distributed among the integers from $-100$ to $100$, then solve the corresponding zero-sum game using the regular shadow vertex method, and record all visited shadow vertices and the corresponding tables. Then, the column player generates a new action and hence produces a new random payoff column whose elements are identically, independently, and uniformly distributed among the integers from $-100$ to $100$. Since the columns of the extended payoff matrix are independently, identically, and uniformly distributed, Assumption 2 is satisfied, and Assumptions 1 and 3 are also satisfied with probability almost 1.

After the new action is generated, we first decide whether re-computation of the security strategy is necessary according to Theorem 1. If so, we use both the regular shadow vertex method and the iterative shadow vertex method to find the new security strategy and record the numbers of pivot steps of both methods. This experiment is run 500 times. Following the same steps, we also test the iterative shadow vertex method in $10 \times 200$, $10 \times 300$, ..., $10 \times 1000$ payoff matrices. The probability of re-computing the security strategy, the average number of pivot steps, and the average number of pivot steps conditioned on re-computation are given in the plots in Fig. 1, where $x$-axis is the size of the action set of player 2.

The top plot of Fig. 1 shows the probability that player 1’s security strategy changes. The red stars are the analytical upper bound on the probability computed according to Theorem 1, and the blue circles are the empirical probability derived from the simulation results. We see that the simulated results match the analytical results. Meanwhile, we also notice that the probability of re-computing security strategy is decreasing with respect to $m$, the size of the action set of player 2, which meets our expectation.
The middle plot of Fig. 1 gives the average number (blue circles) of pivot steps of the iterative shadow vertex method and the appropriately scaled average number (red stars) of pivot steps of the regular shadow vertex method in accordance with the result in Theorem 2. We see that the blue circles is always below the red stars, and both decrease in the same manner as the size of player 2’s action set grows. It matches the result shown in Theorem 2 which indicates that the computational complexity of the iterative shadow vertex method is bounded from above by the appropriately scaled computational complexity of the regular shadow vertex method.

The bottom plot of Fig. 1 shows that the average number (blue circles) of pivot steps of the iterative shadow vertex method is always less than the average number (red stars) of pivot steps of the regular shadow vertex method conditioned on the situation that player 1 needs to re-compute the security strategy, which agrees with the theoretical predictions of Theorem 2.

### 7 Case Study: Urban Security Problem

We now consider a more applied example inspired by the urban security scenario introduced in [17]. The problem in [17] is to determine the optimal placement of police checkpoints on a graph representing the southern part of Mumbai so as to best protect targets. In [17], the urban area is modeled as a graph where edges denote roads, and vertices denote places of interest. This area has several main entrances denoted as source nodes in the graph, and several targets that attackers want to attack. With limited resources, the police need to set up checkpoints on edges to protect the targets. If a checkpoint is in the path of the attackers, the police get a corresponding reward. Otherwise, penalty is issued to the police. In keeping with the theme of this paper, we are interested in fast re-computation of a checkpoint deployment strategy when new targets become of interest to the attackers.

The original graph model in [17] contains 35 nodes and 58 edges. Since we do not have access to the original graph model, a graph with comparable size, 36 nodes and 60 edges, is generated as shown in Fig. 2. We independently, identically, and uniformly choose 3 source nodes (blue circles) and 4 targets (red squares) in the graph. We assume that the attackers use the shortest path to hit the target, and the defenders can use an expected number of 3 edges to set checkpoints, the defenders’ security strategy is to set a checkpoint on the highlighted red edges with equal probability $\frac{3}{7}$ (Color figure online).
edges to set up checkpoints. If a checkpoint is in the path of the attackers, defenders get reward 1, 0 otherwise. There are 37 paths from the source nodes to the target nodes, and the corresponding payoff matrix $G$ is of size $60 \times 37$. Let $\bar{x}_i$ indicate the probability that a checkpoint is set on edge $i$ and $\bar{y}_j$ indicate the probability that attackers choose path $j$. The urban security problem can be modeled as the following maxmin problem.

$$V = \max_{\bar{x} \in \mathbb{R}^{60}} \min_{\bar{y} \in \Delta(P)} \bar{x}^T G \bar{y},$$

s.t. $0 \leq \bar{x} \leq 1$

$$1^T \bar{x} = 3$$

where $P$ is the set of paths that attackers can take. Notice that $\bar{x}_i$ is the probability that a checkpoint is set on edge $i$, but the vector $\bar{x}$ is not a probability vector. The constraint $1^T \bar{x} = 3$ corresponds to the requirement that the defenders can use an expected number of 3 edges. According to the strong duality theorem, it can be transformed to an LP problem

$$V = \max_{\bar{x} \in \mathbb{R}^{60}} \ell,$$

s.t. $G^T \bar{x} \geq \ell 1$

$$0 \leq \bar{x} \leq 1$$

$$1^T \bar{x} = 3$$

The corresponding canonical form is as follows.

$$V = \max_{x \in \mathbb{R}^{60}} \begin{bmatrix} 0 & \ldots & 0 & 1 \end{bmatrix} x,$$

s.t. $\begin{bmatrix} -G^T T & 1 \\ 1^T & 0 \\ -I_{n-1} & 0 \\ I_{n-1} & 0 \end{bmatrix} x \leq \begin{bmatrix} 3G^Te_n \\ 3 \\ 0 \\ 1 \end{bmatrix}$

We use the regular shadow vertex method to solve this problem and find that a security strategy is to set up checkpoints at the red highlighted edges with probability $\frac{3}{7}$, and the value of the game is $\frac{3}{7}$. With 2.4GHz CPU and 4GB memory, the computation time is about 0.35 seconds, which is comparable to the time reported in [17] to compute an equilibrium strategy for a grid of similar size (35 nodes and 58 edges) representing south Mumbai. Notice that the highlighted edges are also the minimum number of edges that can cut all paths (only shortest paths are considered here) from the sources to the targets.

Now, assuming that the attackers randomly choose a new target in the map according to the uniform distribution, we use both the shadow vertex method and the iterative shadow vertex method to update the security strategy. There are two small differences between this situation and that described earlier: (1) $x$ is not a probability vector and (2) each new target will result in several new paths, which adds several columns to the payoff matrix at a time. More precisely, Eq. (26) is changed to $[-g^T T \ 1] x \leq 3g^Te_n$, where $g$ is the newly added payoff columns. Besides these differences, we also see that Assumption 1 is hard to check, and Assumptions 3 and 2 are not satisfied. For Assumption 1, there are 157 constraints and 60 variables in the canonical form, and the number of possible vertices is $(\frac{157}{60}) = 1.46 \times 10^4$. Because of the huge number of possible vertices, it is hard to check whether Assumption 1 is satisfied in this case. For Assumption 3, an alternative strategy is to replace the checkpoint at the fourth bottom edge with a checkpoint at the third bottom edge (count from the left). For Assumption 2, although the source nodes, the old target nodes, and the new target node are chosen
independently, identically, and uniformly, the columns of payoff matrix are not necessarily so. Because for each pair of source and target node, there are generally several paths which are correlated with each other, and the corresponding payoff columns are also correlated. However, these differences and violations of assumptions only affect the probability analysis and the computational complexity analysis of the algorithms. On the other hand, Algorithm 3 and the necessary and sufficient condition of unchanging security strategy in Theorem 1 are in fact applicable as is to this scenario as well.

We randomly choose the new target for 100 times. The probability of re-computing security strategy is 0.94. For the iterative shadow vertex method, the average number of pivot steps is 7.40, and the average number of pivot steps conditioned on the re-computation of security strategy is 7.80. For the regular shadow vertex method, the average number of pivot steps is 16.05, and the average number of pivot steps conditioned on the re-computation of security strategy is 16.73. Although Assumptions 2 and 3 are violated and Assumption 1 is not guaranteed, the iterative shadow vertex method still cuts more than half of the average number of pivot steps of the regular shadow vertex method.

8 Conclusion and Future Work

This paper studies how to efficiently update the saddle-point strategy of one player in a matrix game when the other player can add new actions in the game. We provide an iterative shadow vertex method to solve this problem and show that the computational complexity is strictly less than that of the regular shadow vertex method. Moreover, this paper also presents a necessary and sufficient condition that a new saddle-point strategy is needed and analyzes the probability of re-computing the saddle-point strategy. Our simulation results demonstrate the main results.

A direct extension of the problem in this paper is its dual problem, i.e., the case when player 1 has a growing action set. In this case, the corresponding LP has new variables whose dual problem is exactly the same problem as studied in this paper. We can use the iterative shadow vertex method to solve its dual problem first and then figure out the optimal solution from the optimal solution of its dual problem. A further extension is the case when both players have growing action sets. A proposed direction is to deal with player 2’s new action first and then deal with player 1’s action set.

Appendix

Proof of Lemma 1

Assume that Eq. (15) holds. We know that $A_{\Omega_0^i} x_0 = b_{\Omega_0^i} \geq A_{\Omega_0^i} x$ for any feasible $x$, which implies that $w^T x_0 \geq w^T x$ for any feasible $x$.

Now assume that $w^T x_0 \geq w^T x$ for any feasible $x$. According to the inhomogeneous Farkas theorem [10], we derive that $w = \sum_{i=1}^m \rho_i A_i^T$ for some nonnegative vector $\rho$ satisfying $\sum_{i=1}^m \rho_i b_i \leq w^T x_0 = \sum_{i=1}^m \rho_i A_i x_0$. It implies that $\sum_{i \notin \Omega_0} \rho_i (b_i - A_i x_0) \leq 0$. Since $\rho_i \geq 0$ and $b_i - A_i x_0 > 0$ for $i \notin \Omega_0$, the inequality holds only if $\rho_i = 0$ for $i \notin \Omega_0$, and Eq. (15) is true.
Proof of Proposition 1

1) ⇒ 2): It is clear that if $\hat{x}$ is a shadow vertex, its projection lies in the boundary of $\Gamma(X)$.

2) ⇒ 3): Let $\Gamma(\hat{x}) \in \partial \Gamma(X)$. Because of the convexity of the $\Gamma(X)$, there must exist a $w \in \text{span}(u, c) \setminus \{0\}$ such that $w^T \Gamma(\hat{x}) \geq w^T \Gamma(x)$ for any $x \in X$. Meanwhile, we know that $x - \Gamma(x) \perp \text{span}(u, c)$, and hence, $w^T x = w^T \Gamma(x)$ for any $x \in X$. Therefore, we have $w^T \hat{x} \geq w^T x$ for any $x \in X$.

3) ⇒ 1): Now let’s assume that 3) is true. Because $x - \Gamma(x) \perp \text{span}(u, c)$ and $w \in \text{span}(u, c)$, we know that $\Gamma(\hat{x})$ is the maximal relative to $w$ for $x \in \Gamma(X)$. Therefore, $\Gamma(\hat{x})$ is in the boundary of the shadow $\Gamma(X)$. Since $\Gamma(X)$ is a two-dimensional polygon, if $\Gamma(\hat{x})$ is not a vertex, then it must lies inside an edge.

Together with the fact that $\Gamma(\hat{x})$ is the optimum w.r.t $w$, we know that $w$ is orthogonal to the edge, and there exists a $v \in \text{span}(u, c) \setminus \{0\}$ such that $w \perp v$ and $\Gamma(\hat{x})$ is the maximal relative to $w + \epsilon v$ for any $x \in \Gamma(X)$ if and only if $\epsilon = 0$. Since $x - \Gamma(x) \perp \text{span}(u, c)$ and $w + \epsilon v \in \text{span}(u, c)$, we know that $\hat{x}$ is also maximal relative to $w + \epsilon v$ if and only if $\epsilon = 0$ for any $x \in X$.

Let $\Omega$ be the active constraint set when $x = \hat{x}$. Assumption 1 indicates that there are $n$ elements in $\Omega$. Since $\hat{x}$ is maximal relative to $w$, according to Theorem 1, $w = \sum_{i=1}^{n} \rho_i A^{T}_{\Omega_i} \neq 0$ for $\rho \geq 0$. Let $v = \sum_{i=1}^{n} \alpha_i A^{T}_{\Omega_i}$, and hence, $w + \epsilon v = \sum_{i=1}^{n} (\rho_i + \epsilon \alpha_i) A^{T}_{\Omega_i}$. $\hat{x}$ is not maximal relative to $w + \epsilon v$ for $\epsilon > 0$ implies that there exists an $l$ such that $\rho_l + \epsilon \alpha_l < 0$ for any $\epsilon > 0$, and from the continuity of the function, we see that $\rho_l + \alpha_l = 0$. Similarly, $\hat{x}$ is not maximal relative to $w + \epsilon v$ for $\epsilon < 0$ implies that there exists a $k$ such that $\rho_k + \epsilon \alpha_k < 0$ for any $\epsilon < 0$, and the continuity of the function implies that $\rho_k + \alpha_k = 0$. Moreover, together with the fact that $\rho_i + \epsilon \alpha_i$ is a linear function of $\epsilon$ for $i = l, k$, we see that $l \neq k$.

Therefore, we know that at least 2 elements of $\rho$ is 0, and we have

$$w = \beta_1 u + \beta_2 c = \sum_{i=1, i \neq k, l}^{n} \rho_i A^{T}_{\Omega_i}.$$  \hspace{1cm} (38)

It means that there exist linearly dependent $n$ elements in $\{A_{\Omega_1}, \ldots, A_{\Omega_n}, u, c\}$, which contradicts Assumption 1, and hence, $\Gamma(\hat{x})$ is a vertex of $\Gamma(X)$.

Results Related to Theorem 1

Lemma 4 Suppose Assumptions 1, 3 hold. The unique optimal solution $x^*$ of the old LP (10, 11) is not optimal any more in the new LP (25, 26) if and only if the new constraint (27) is an active constraint for any optimum of the new LP (25, 26), i.e., $[g_n - g_1 g_n - g_2 \ldots g_n - g_{n-1} 1] x^+ = g_n$, where $x^+$ is any optimum of the new LP (25, 26).

Proof Suppose $x^*$ is not an optimum of the new LP. According to Theorem 1, we have $\hat{A}_{m+1} x^+ > \hat{b}_{m+1}$. If there exists an optimum $x^+$ of the new LP such that $\hat{A}_{m+1} x^+ < \hat{b}_{m+1}$. then there must exist an $x = \alpha x^+ + (1 - \alpha) x^*$ such that $\hat{A}_{m+1} x = \hat{b}_{m+1}$. It is easy to verify that $x$ is a feasible solution of the new LP. Since $x^+$ is an optimum of the new LP, we have

$$c^T x^+ \geq c^T x.$$  \hspace{1cm} (39)

Meanwhile, $x^*$ is the unique optimum of old LP implies that $c^T x^+ < c^T x^*$, and we have

$$e^T x = \alpha c^T x^+ + (1 - \alpha) c^T x^* > e^T x^+$$
which contradicts Eq. (39). Therefore, there does not exist an optimum $x^+$ such that $\hat{A}_{m+1}x^+ < \hat{b}_{m+1}$. Together with the fact that an optimal solution is feasible, we conclude that $\hat{A}_{m+1}x^+ = \hat{b}_{m+1}$ for any optimum of the new LP problem.

For the other direction, suppose the new constraint is active for any optimum of new LP (25, 26). If the old optimum $x^*$ is an optimum of the new LP, the new constraint (27), i.e., constraint $m + 1$, is active. Together with the other $n$ active constraints which are included in the old constraints, there are $n + 1$ active constraints. This contradicts Assumption 1. Therefore, $x^*$ is not an optimum of the new LP.

For the simplicity of the following two lemmas, we let

$$\Omega_1 = \{i_1, \ldots, i_\ell, \kappa_1, \ldots, \kappa_{n-\ell}\} \quad (40)$$

$$\Omega_2 = \{j_1, \ldots, j_\ell, \kappa_1, \ldots, \kappa_{n-\ell}\} \quad (41)$$

for some $\ell = 1, \ldots, n$, where $\kappa_1, \ldots, \kappa_{n-\ell} \in \{m + 2, \ldots, m + 1 + n\}$ are the indices of probability constraints, and $i_1, \ldots, i_\ell, j_1, \ldots, j_\ell \in \{1, \ldots, m + 1\}$ are the indices of normal constraints.

**Lemma 5** Suppose Assumptions 1 and 2 hold. Any $n$ element subset $\Delta \subset \{1, \ldots, m + 1 + n\}$ that contains the same probability constraints has the same probability to be the optimal active constraint set of the new LP problem (25, 26). In other words,

$$P(\Omega_1 \text{ is the optimal active constraint set}) = P(\Omega_2 \text{ is the optimal active constraint set}). \quad (42)$$

where $\Omega_1$ and $\Omega_2$ are defined as in (40) and (41), respectively.

**Proof** Since the payoff columns are independently and identically distributed, $\hat{A}_1, \ldots, \hat{A}_{m+1}$ are also independently and identically distributed. Therefore, The probability that the convex cone of $\hat{A}_{i_1}, \ldots, \hat{A}_{i_\ell}, \hat{A}_{\kappa_1}, \ldots, \hat{A}_{\kappa_{n-\ell}}$ contains $c$ is the same as the probability that the convex cone of $\hat{A}_{j_1}, \ldots, \hat{A}_{j_\ell}, \hat{A}_{\kappa_1}, \ldots, \hat{A}_{\kappa_{n-\ell}}$ contains $c$. Together with Lemma 1, we see that Eq. (43) is true. \hfill \square

**Related Results in the Proof of Theorem 2**

**Lemma 6** Consider the new LP problem (25, 26), and suppose Assumptions 1 and 2 hold. Any $n$ element subset $\Delta \subset \{1, \ldots, m + 1 + n\}$ that contains the same probability constraints has the same probability to form a shadow vertex of the new LP problem (25, 26). In other words,

$$P(\Omega_1 \text{ forms a shadow vertex}) = P(\Omega_2 \text{ forms a shadow vertex}), \quad (43)$$

where $\Omega_1$ and $\Omega_2$ are defined as in (40) and (41), respectively.

**Proof** Since the payoff columns are independently and identically distributed, $\hat{A}_1, \ldots, \hat{A}_{m+1}$ are also independently and identically distributed. Therefore, The probability that the convex cone of $\hat{A}_{i_1}, \ldots, \hat{A}_{i_\ell}, \hat{A}_{\kappa_1}, \ldots, \hat{A}_{\kappa_{n-\ell}}$ intersects with $\text{span}(u, c)$ is the same as the probability that the convex cone of $\hat{A}_{j_1}, \ldots, \hat{A}_{j_\ell}, \hat{A}_{\kappa_1}, \ldots, \hat{A}_{\kappa_{n-\ell}}$ intersects with $\text{span}(u, c)$. Together with Theorem 1, we see that Eq. (43) is true. \hfill \square
References

1. Agrawal S, Wang Z, Ye Y (2014) A dynamic near-optimal algorithm for online linear programming. Oper Res 62(4):876–890
2. Başar T, Olsder GJ (1998) Dynamic noncooperative game theory. SIAM
3. Bopardikar SD, Langbort C (2016) On incremental approximate saddle-point computation in zero-sum matrix games. Automatica 69:150–156
4. Borgwardt KH (1982) The average number of pivot steps required by the simplex-method is polynomial. Math Methods Oper Res 26(1):157–177
5. Borgwardt KH (2012) The simplex method: a probabilistic analysis, vol 1. Springer, Berlin
6. Dantzig G (2016) Linear programming and extensions. Princeton University Press, Princeton
7. Devanur NR, Hayes TP (2009) The adwords problem: online keyword matching with budgeted bidders under random permutations. In: Proceedings of the 10th ACM conference on electronic commerce. ACM, pp 71–78
8. Eghbali R, Fazel M (2016) Worst case competitive analysis of online algorithms for conic optimization. arXiv:161100507
9. Feldman J, Henzinger M, Korula N, Mirrokni V, Stein C (2010) Online stochastic packing applied to display ad allocation. In: European symposium on algorithms. Springer, 2010, pp 182–194
10. Iouditski A (2016) Course note of convex optimization i: introduction. http://ljk.imag.fr/membres/Anatoli.Iouditski/cours/convex/chapitre_2.pdf
11. Kleinberg R (2005) A multiple-choice secretary algorithm with applications to online auctions. In: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, pp 630–631
12. Liebling TM, Henn R, Künzi H, Schubert H (1972) On the number of iterations of the simplex method. Methods Oper Res 2(ROSO–CONF–1972–001):248–264
13. Lindberg PO (1981) A note on random LP-problems. Department of Mathematics, Royal Institute of Technology, Stockholm
14. Molinaro M, Ravi R (2013) The geometry of online packing linear programs. Math Oper Res 39(1):46–59
15. Schmidt WM (1968) Some results in probabilistic geometry. Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete 9(2):158–162
16. Spielman DA, Teng SH (2004) Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. J ACM: JACM 51(3):385–463
17. Tsai J, Yin Z, Kwak Jy, Kempe D, Kiekintveld C, Tambe M (2010) Urban security: game-theoretic resource allocation in networked physical domains. In: National conference on artificial intelligence (AAAI)