Computing mixed strategies equilibria in presence of switching costs by the solution of nonconvex QP problems

G. Liuzzi · M. Locatelli · V. Piccialli · S. Rass

Received: date / Accepted: date

Abstract In this paper we address a game theory problem arising in the context of network security. In traditional game theory problems, given a defender and an attacker, one searches for mixed strategies which minimize a linear payoff functional. In the problem addressed in this paper an additional quadratic term is added to the minimization problem. Such term represents *switching costs*, i.e., the costs for the defender of switching from a given strategy to another one at successive rounds of the game. The resulting problem is a nonconvex QP problem with linear constraints. We prove that, though simple when one of the two terms in the objective function is missing, the problem is, in general, a NP-hard one. The proof is based on a reduction from the decision version of the clique problem. Next, we discuss different reformulations and bounds for such problem. Finally, we show, through an extensive set of computational experiments, that the most promising approach to tackle this problem appears to be a branch-and-bound approach where a predominant role is played by an optimality-based domain reduction, based on the multiple solutions of LP problems at each node of the branch-and-bound tree. The computational cost per node is, thus, high but this is largely compensated by the rather small size of the corresponding tree.

Keywords Game Theory · Nonconvex Quadratic Programming Problems · Branch-and-Bound · Bound-Tightening

G. Liuzzi
Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" (IASI)
Consiglio Nazionale delle Ricerche (CNR)
Via dei Taurini 19, 00185 Rome - Italy
Tel.: +0039 06 4993 7129
E-mail: giampaolo.liuzzi@cnr.iasi.it

M. Locatelli
Università degli Studi di Parma
Parco Area delle Scienze, 181/A - I 43124 Parma
E-mail: marco.locatelli@unipr.it

V. Piccialli
DICII - University of Rome Tor Vergata
via del Politecnico 1
00133 Roma
E-mail: veronica.piccialli@uniroma2.it

S. Rass
Universitat Klagenfurt, Institute of Applied Informatics, System Security Group,
Klagenfurt, Austria,
E-mail: stefan.rass@aau.at

1 Introduction

Consider a finite two-person zero-sum game $\varGamma$, composed from a player set $N = \{1, 2\}$, each member thereof having a finite strategy space S_1, S_2 associated with it, and a utility function $u_i : S_1 \times S_2 \to \mathbb{R}$ for all $i \in N$. We assume a zero-sum competition, making $u_2 := -u_1$ hereafter. The game is then the triple $\varGamma = (N, \mathcal{S} = \{S_1, S_2\}, H = \{u_1, -u_1\})$, and is most compactly represented by giving only the payoff function u_1 in matrix form (since the strategy spaces are finite) as

$$\mathbf{A} \in \mathbb{R}^{|S_1|\times|S_2|} = (u_1(x, y))_{(x,y)\in S_1\times S_2}\,.$$

An equilibrium in $\varGamma$ is a simultaneous optimum for both players w.r.t. u_1. Assuming a maximizing first player, an equilibrium is a pair (x^*, y^*) satisfying the saddle-point condition

$$u_1(x, y^*) \le u_1(x^*, y^*) \le u_1(x^*, y) \quad \forall (x, y) \in S_1 \times S_2.$$

It is well known that many practical games do not have such an equilibrium point; as one of the simplest instances, consider the classical rock-scissors-paper game, represented by the payoff matrix

$$\begin{array}{r c} & \begin{array}{ccc} \text{rock} & \text{scissors} & \text{paper} \end{array} \\ \begin{array}{r} \text{rock} \\ \text{scissors} \\ \text{paper} \end{array} & \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}. \end{array}$$

This game has no equilibria in pure strategies: any fixed choice of rock, scissors or paper would imply a constant loss for the first player (and likewise for the second player). This means that player 1 is forced to *randomize* its actions in every round of the game, and this concept leads to the idea of mixed extensions of a game, which basically changes the above optimization problem into one over the convex hulls $\varDelta(S_1), \varDelta(S_2)$ of the action spaces, rather than the finite sets S_1, S_2. An element of $\varDelta(S_i)$ is then a probability distribution over the elements of the support S_i, and prescribes to pick a move at random whenever the game is played.

The game rewards its players after each round, and upon every new round, both players are free to choose another element from their action space at random. Implicitly, this choice is without costs, but what if not? Many real life instances of games *do incur* a cost for changing one's action from $a_1 \in S_1$ in the first to some distinct $a_2 \in S_1$ in the next round. The area of system security [1,19] offers rich examples of such instances, such as (among many):

- Changing passwords [14]: if the currently chosen password is p_1 and we are obliged to pick a fresh password (say, different from the last couple of passwords that we had in the past), the use of the new password $p_2 \neq p_1$ induces quite some efforts, as we have to memorize the password, while choosing it as hard as possible to guess.
- Patrolling and surveillance [2,13]: consider a security guard on duty to repeatedly check a few locations, say A, B, . . . , E, which are connected at distances as depicted in Fig. 1. This is a chasing-evading game with the guard acting as player 1 against an intruder being player 2, and with the payoff function u being an indicator of whether the guard caught the intruder at location $i \in \{\text{A},\ldots,\text{E}\}$, or whether the two missed each other. This is yet another instance of a game with all equilibria in mixed strategies, but with the unpleasant side-effect for the guard that gets the prescription to randomly spot check distant locations to "play" the equilibrium $\mathbf{x}^*$, the guard would have to move perhaps long distances between the locations. For example, if it is at A in round 1 and the next sample from the random distribution $\mathbf{x}^* \in \varDelta(\{\text{A},\ldots,\text{E}\})$ tells to check point E next, the shortest path would be of length $1 + 3 + 2 = 6$ over C. Starting from A, however, it would be shorter and hence more convenient for the guard to check location B first along the way, but this would mean deviating from the equilibrium! A normal game theoretic equilibrium calculation does not consider this kind of investment to change the current strategy. This may not even count as bounded rationality, but simply as acting "economic"

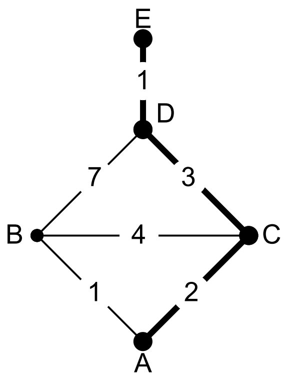


Fig. 1: Example of spot checking game on a graph

from the guard's perspective. But acting economically here is then not governed by a utility maximizing principle, but rather by a cost minimization effort.
- Generalizing the patrolling game example, the issue applies to all sorts of moving target defense: for example, changing the configuration of a computer system so as to make it difficult for an attacker to break in, often comes with high efforts and even risks for the defending player 1 (the system administrator), since it typically means taking off machines from the network, reconfiguring them to close certain vulnerabilities, and then putting them back to work hoping that everything restarts and runs smoothly again. A normal game theoretic model accounts only for the benefits of that action, but not for the cost of *taking* the action.

Including the cost to switch from one action to the next is more complicated than just assigning a cost function $c : S_1 \to \mathbb{R}$ and subtracting this from the utilities to redefine them as $u_1'(i,j) = u_1(i,j) - c(i)$, since the cost to play a_i will generally depend on the previous action a_j played in the previous round.

We can model this sort of payoff by another function $s : S_1 \times S_1 \to \mathbb{R}$ that we call the *switching cost*. The value of $s(i,j)$ is then precisely the cost incurred to change the current action $i \in S_1$ into the action $j \in S_1$ in the next round of the game. Intuitively, this adds another payoff dimension to the game, where a player, w.l.o.g. being player 1 in the following, plays "against itself", since the losses are implied by its own behavior. While the expected payoffs in a matrix game $\mathbf{A}$ under mixed strategies $\mathbf{x} \in \Delta(S_1), \mathbf{y} \in \Delta(S_2)$ are expressible by the bilinear functional $\mathbf{x}^T\mathbf{A}\mathbf{y}$, the same logic leads to the hypothesis that the switching cost should on average be given by the quadratic functional $\mathbf{x}^T\mathbf{S}\mathbf{x}$, where the switching cost matrix is given, like the payoff matrix above, as

$$\mathbf{S} \in \mathbb{R}^{|S_1|\times|S_1|} = (s(x,y))_{(x,y)\in S_1\times S_1}.$$

This intuition is indeed right [15], but for a rigorous problem statement, let us briefly recap the derivation given independently later by [24] to formally state the problem.

2 The problem

Let the game come as a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ with equilibrium $(\mathbf{x}^*, \mathbf{y}^*)$, and let it be repeated over the time $t \in \mathbb{N}$. At each time t, let $X_t \sim \mathbf{x}^*$ be the random action sampled from the equilibrium distribution over the action space (with $\mathbf{x}^*$ being the optimal distribution). By the implicit yet standard assumption of stochastic independence between two repetitions of the game, we have $\Pr(X_{t-1} = i, X_t = j) = \Pr(X_{t-1} = i) \cdot \Pr(X_t = j)$, so that any future system state remains equally predictable whether or not the current state of the system is known. Hence, the switching cost can be written as

$$\begin{aligned} s(X_{t-1}, X_t) &= \sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij} \cdot \Pr(X_{t-1} = i, X_t = j) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij} \cdot \Pr(X_{t-1} = i) \cdot \Pr(X_t = j) = \mathbf{x}^T\mathbf{S}\mathbf{x}. \end{aligned}$$

With this, player 1's payoff functional becomes vector-valued now as

$$\mathbf{u}_1 : \Delta(S_1) \times \Delta(S_2) \to \mathbb{R}^2, \quad (\mathbf{x},\mathbf{y}) \mapsto \begin{pmatrix} u_1(\mathbf{x},\mathbf{y}) = \mathbf{x}^T\mathbf{A}\mathbf{y} \\ s(\mathbf{x},\mathbf{y}) = \mathbf{x}^T\mathbf{S}\mathbf{x} \end{pmatrix}, \tag{1}$$

and the game is multi-objective for the first player. As we are interested mostly in the best behavior for player 1 and the analysis would be symmetric from player 2's perspective, we shall not explore the view of the second player hereafter.

Remark 1 The game could be equally well multi-objective for the second player too, and in fact a practical instance of such a situation may also come from security: it could be in an adversary's interest to "keep the defender busy", thus causing much friction by making the defender move fast from one place to the other. This is yet just another instance of a denial-of-service attack, to which such a game model would apply.

For the sake of computing a multi-objective equilibrium, more precisely a Pareto-Nash equilibrium, the algorithm in [16] based on the method laid out in [11] proceeds by scalarizing (1) by choice of some $\alpha \in (0, 1)$, to arrive at the real-valued goal function

$$\alpha \cdot \mathbf{x}^T\mathbf{A}\mathbf{y} + (1-\alpha) \cdot \mathbf{x}^T\mathbf{S}\mathbf{x},$$

for the first player to optimize. Now, the usual way from here to an optimization problem for player 1 involving a rational opponent applies as for standard matrix games [15]: let $\mathbf{e}_i \in \mathbb{R}^m$ be i-th unit vector, then $\arg\max_{\mathbf{y}\in\Delta(S_2)}(\mathbf{x}^T\mathbf{A}\mathbf{y}) = \arg\max_i(\mathbf{x}^T\mathbf{A}\mathbf{e}_i)$. Substituting $v := \max_i(\alpha \cdot \mathbf{x}^T\mathbf{A}\mathbf{e}_i + (1-\alpha)\cdot\mathbf{x}^T\mathbf{S}\mathbf{x})$, the resulting problem becomes

$$\left.\begin{array}{rrl} & v & \to \min \\ \text{subject to} & v & \geq (1-\alpha)\cdot\mathbf{x}^T\mathbf{S}\mathbf{x} + \alpha\mathbf{x}^T\mathbf{A}\mathbf{e}_i \quad \text{for } i = 1,\ldots,m; \\ & \sum_{j=1}^{n} x_j & = 1; \\ & x_i & \geq 0, \quad \text{for } i = 1,\ldots,n. \end{array}\right\} \tag{2}$$

which is almost the familiar optimization program to be solved for a Nash equilibrium in a finite matrix game. It differs from the well known linear program only in the quadratic term, and in fact, the equilibrium problem for matrix games is recovered by substituting $\alpha = 1$ in (2). The problem would again become trivial for $\alpha = 0$, since in that case, only the switching cost matters and hence every degenerate distribution corresponding to a strategy $i \in S_1$ that never changes is directly an equilibrium. Excluding the standard equilibria obtained at $\alpha = 1$ and the meaningless results expected for $\alpha = 0$, problem (2) is interesting only for values of α strictly between 0 and 1.

Note that the matrix $\mathbf{S}$ in the quadratic term will (in most cases) have a zero diagonal, be indefinite and not symmetric in general (patrolling game example given above already exhibits a variety of counterexamples leading to nonsymmetric distance matrices $\mathbf{S}$ if the graph is directed). Of course, symmetry of $\mathbf{S}$ can be easily recovered, so in what follows we will assume that $\mathbf{S}$ is symmetric.

Remark 2 (On Alternatives to Solve the Switching Cost Problem) Game models that reward players based on their past actions are stochastic games. Extending equilibria to this class of games is possible [18], yet can render such models practically infeasible to set up (the model then contains the specification of a stochastic chain whose states equal different payoff structures, and the modeler is required to specify transition probabilities between different such games; an information that people in practice may lack or find difficult to reliably give).

The dependence of next actions on past ones extends to other scenarios too: for example, if the game is about coordination in wireless settings (e.g., collaborative drones), the players, e.g., drones, share a common communication channel. Every exchange of information occupies that channel for a limited period of time, thus constraining what the other players can do at the moment. Such effects can be described by stochastic games, but depending on how far the effect reaches in the future, backward inductive solution methods may become computationally infeasible [8]; likewise, extending the strategy space to plan ahead a fixed number of k steps (to account for one strategy determining the next k repetitions of the game) may exponentially enlarge the strategy space (by a factor of $2^{O(k)}$, making the game infeasible to analyze if k is large). Games with switching cost offer a neat bypass to that trouble: if an action is such that it occupies lots of resources for a player, thus preventing it from taking further moves in the next round of the game, we can express this as a switching cost. Assume, for instance, that an action in a game Γ is such that the player is blocked for the next k rounds, then the switching cost is k-times the expected utility $\overline{u}$ (with the expectation taken over the equilibrium distribution played by the participants) that these next k rounds would give. Virtually, the situation is thus like if the player would have paid the total average gain over the next rounds where it is forced to remain idle (thus gaining nothing):

$$\overline{u} \quad \underbrace{-k\cdot\overline{u}}_{\text{switching cost}} \quad + \underbrace{\overline{u}+\cdots+\overline{u}}_{\substack{\text{virtual payoffs}\\ \text{over } k \text{ rounds}}} = \overline{u} + \underbrace{0+0+\ldots+0}_{\substack{\text{practical payoffs}\\ \text{by being idle}\\ \text{for } k \text{ rounds}}} \tag{3}$$

Expression (3) will in practice be only an approximate identity, since we assumed that the game, viewed as a stochastic process, has already converged to stationarity (so that the equilibrium outcome $\overline{u}$ is actually rewarded). The speed of convergence, indeed, can itself be of interest to be controlled in security applications using moving target defenses [24]. The crucial point of modeling a longer lasting effect of the current action like described above, however, lies in the avoidance of complexity: expression (3) has no issues with large k, while more direct methods of modeling a game over k rounds, or including a dependency on the last k moves, is relatively more involved (indeed, normal stochastic games consider a first-order Markov chain, where the next state of the game depends on the last state; the setting just described would correspond to an order k chain, whose conversion into a first order chain is also possible, but complicates matters significantly).

2.1 Paper Outline

The paper is structured as follows. In Section 3 we show that the problem to be solved is a NP-hard one through a reduction from the decision version of the clique problem. In Section 4 we discuss different reformulations (bilinear, MILP) of the problem and possible bounds for it. In Section 5 we present a (spatial) branch-and-bound approach for the problem, putting a particular emphasis on the bound-tightening procedure, which turns out to be the most effective to attack it. In Section 6 we present some computational experiments. We first describe the set of test instances. Next, we discuss the performance of existing solvers over these instances. Finally, we present and comment the computational results attained by the proposed approach.

3 Complexity Result

In this section we consider the complexity of the problem to be solved and we show that its corresponding decision problem is NP-complete. Let

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{S}\mathbf{x} + \max_{k=1,\dots,m} \mathbf{A}_k^T\mathbf{x}, \tag{4}$$

where $S_{ii} = 0$ for each $i = 1,\dots,n$ and $S_{ij} \geq 0$ for each $i \neq j$, while $\mathbf{A}_k \geq \mathbf{0}$, $k = 1,\dots,m$. Moreover, let

$$\Delta_n = \{\mathbf{x} \in \mathbb{R}^n_+ \ : \ \mathbf{e}^T\mathbf{x} = 1\}$$

be the n-dimensional unit simplex. After incorporating α and $(1-\alpha)$ respectively into $\mathbf{S}$ and $\mathbf{A}_k$, $k = 1,\dots,m$, problem 2 introduced in Section 2 is equivalent to $\min_{\mathbf{x}\in\Delta_n} f(\mathbf{x})$. Then, we would like to establish the complexity of the following decision problem:

$$\text{Given a constant } \xi \geq 0 \quad \exists\, \mathbf{x} \in \Delta_n \ : \ f(\mathbf{x}) \leq \xi? \tag{5}$$

As already remarked in Section 2, the problem would be trivial for the quadratic part of f alone (the minimum of the quadratic form in f over Δ_n is equal to 0 and is attained at each vertex of the unit simplex). It would also be polynomially solvable for the piece-wise linear part of f alone (one only needs to solve a linear problem). However, it turns out that the overall problem is difficult, as proved in the following theorem.

Theorem 31 *The decision problem (5) is NP-complete.*

Proof. We prove the result by providing a polynomial transformation of the max clique decision problem, i.e., given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, does there exist a clique C in G with cardinality at least k? We define the following instance of the decision problem (5). Let

$$S_{ij} = \begin{cases} 0 & \text{if } i = j \text{ or } (i,j) \in E \\ n^4 & \text{otherwise.} \end{cases}$$

Moreover, let $m = n$ and for each $k = 1,\dots,n$ let $\mathbf{A}_k = \mathbf{e}_k$, where $\mathbf{e}_k$ is the vector with all components equal to 0, except the k-th one, which is equal to 1. Stated in another way, the piece-wise linear part is $\max_{k=1,\dots,n} x_k$. Finally, let $\xi = \frac{1}{k}$. We claim that the minimum value of f over Δ_n is not larger than $\xi = \frac{1}{k}$ if and only if G contains a clique with cardinality k. The *if* part is very simple. Indeed, let us consider the feasible solution $x_i = \frac{1}{k}$ if $i \in C$, where C is a clique of cardinality k, and let $x_i = 0$ otherwise. Then, the value of f at this point is equal to $\frac{1}{k}$. Indeed , the value of the quadratic part is 0, while the value of the piece-wise linear part is $\frac{1}{k}$. The proof of the *only if* part is a bit more complicated. We would like to prove that, in case no clique with cardinality at least k exists, then the minimum value of f over the unit simplex is larger than $\frac{1}{k}$. Let us denote by $\mathbf{x}^*$ the minimum of f over the unit simplex. Let

$$K = \mathit{supp}(\mathbf{x}^*) = \{i \ : \ x_i^* > 0\},$$

and let C be the maximum clique over the sub-graph induced by K, whose cardinality is at most $k-1$. We first remark that if, for some $i \in K \setminus C$, it holds that

$$x_i^* \geq \frac{1}{n^2} \quad \text{and} \quad \sum_{j\in C \ : \ (i,j)\notin E} x_j^* \geq \frac{1}{n^2},$$

then the quadratic part contains the term

$$n^4 x_i^* \left(\sum_{j\in C \ : \ (i,j)\notin E} x_j^* \right) \geq 1,$$

which concludes the proof. Therefore, we assume that either $x_i^* < \frac{1}{n^2}$ or

$$\sum_{j\in C\ :\ (i,j)\notin E} x_j^* < \frac{1}{n^2}. \tag{6}$$

Now, let

$$K_1 = \left\{ i\ :\ i \in K \setminus C \text{ and } x_i^* \geq \frac{1}{n^2} \right\}.$$

If $\exists\ k_1, k_2 \in K_1$ and $(k_1, k_2) \notin E$, then $n^4 x_{k_1}^* x_{k_2}^* \geq 1$, which concludes the proof. Then, we assume that for each $k_1, k_2 \in K_1$, $(k_1, k_2) \in E$, i.e., K_1 itself is a clique. Now let us consider the following subset of C

$$C_1 = \{i \in C\ :\ (i,k) \notin E \text{ for at least one } k \in K_1\}.$$

It must hold that $|C_1| \geq |K_1|$. Indeed, if $|C_1| < |K_1|$, then $(C \setminus C_1) \cup K_1$ is also a clique with cardinality larger than C, which is not possible in view of the fact that C has maximum cardinality. Then, in view of (6) we have that

$$x_i^* < \frac{1}{n^2} \quad \forall\ i \in C_1,$$

and, moreover, by definition of K_1, we also have

$$x_i^* < \frac{1}{n^2} \quad \forall\ i \in K \setminus (K_1 \cup C).$$

Since $|C_1| \geq |K_1|$, we have that

$$T = \left\{ i \in K\ :\ x_i^* \geq \frac{1}{n^2} \right\}$$

is such that $|T| \leq |(K \setminus (K_1 \cup C)) \cup C_1| \leq |C| \leq k-1$. Thus, taking into account that

$$\sum_{i\in K\setminus T} x_i^* < \frac{1}{n},$$

we must have that

$$\sum_{i\in T} x_i^* > 1 - \frac{1}{n},$$

and, consequently, taking into account that $|T| \leq k-1$, for at least one index $j \in T$ it must hold that

$$x_j^* > \frac{1-\frac{1}{n}}{k-1} \geq \frac{1}{k},$$

so that the piece-wise linear part of f is larger than $\frac{1}{k}$, which concludes the proof. □

4 A review of different reformulations and bounds

In the next subsections we will briefly present and discuss different reformulations and bounds that we tested to attack the problem of minimizing the objective function (4) over the unit simplex. Later on, in Section 5 we will discuss in more detail the reformulation and the related bounding procedure which turned out to be the best one in practice. Before proceeding, we remark that, by a standard procedure, the convex piece-wise linear term of the objective function can be replaced by a single variable with the addition of linear constraints, i.e.,

$$\begin{aligned} \min\ & \tfrac{1}{2}\mathbf{x}^T\mathbf{S}\mathbf{x} + v \\ & v \geq \mathbf{A}_j^T\mathbf{x} \quad j = 1,\ldots,m \\ & \mathbf{x} \in \Delta_n, \end{aligned} \tag{7}$$

where we recall that Δ_n denotes the n-dimensional unit simplex.

4.1 Bilinear reformulation

A simple reformulation of problem (7), with bilinear objective function and linear constraints, is the following:

$$\begin{aligned} &\min \tfrac{1}{2}\textstyle\sum_{i=1}^{n} x_i y_i + v \\ &v \geq \mathbf{A}_j^T \mathbf{x} \qquad j = 1, \ldots, m \\ &y_i = \mathbf{S}_i \mathbf{x} \qquad i = 1, \ldots, n \\ &\mathbf{x} \in \Delta_n, \end{aligned} \tag{8}$$

where $\mathbf{S}_i$ denotes the i-th row of matrix $\mathbf{S}$. If lower bounds ℓ_{x_i}, ℓ_{y_i} and upper bounds u_{x_i}, u_{y_i} are available for x_i and y_i, respectively, then the well known McCormick underestimating function (see [12])

$$\max \{\ell_{x_i} y_i + \ell_{y_i} x_i - \ell_{x_i} \ell_{y_i}, u_{x_i} y_i + u_{y_i} x_i - u_{x_i} u_{y_i}\},$$

can be employed to limit from below the bilinear term $x_i y_i$ over the rectangle $[\ell_{x_i}, u_{x_i}] \times [\ell_{y_i}, u_{y_i}]$. In fact, it turns out that McCormick underestimating function is the convex envelope of the bilinear term over the given rectangle. Then, after introducing the additional variables f_i, we have that the optimal value of the following LP problem is a lower bound for problem (7) when variables x_i and y_i, $i = 1, \ldots, n$, are bounded to belong to the intervals $[\ell_{x_i}, u_{x_i}]$ and $[\ell_{y_i}, u_{y_i}]$, respectively:

$$\min \frac{1}{2} \sum_{i=1}^{n} f_i + v \tag{9a}$$

$$\mathbf{x} \in \Delta_n \tag{9b}$$

$$\ell_{x_i} \leq x_i \leq u_{x_i} \quad i = 1, \ldots, n \tag{9c}$$

$$v \geq \mathbf{A}_j^T \mathbf{x} \quad j = 1, \ldots, m \tag{9d}$$

$$y_i = \mathbf{S}_i \mathbf{x} \quad i = 1, \ldots, n \tag{9e}$$

$$\ell_{y_i} \leq y_i \leq u_{y_i} \quad i = 1, \ldots, n \tag{9f}$$

$$f_i \geq \ell_{y_i} x_i + \ell_{x_i} y_i - \ell_{x_i} \ell_{y_i} \quad i = 1, \ldots, n \tag{9g}$$

$$f_i \geq u_{x_i} y_i + u_{y_i} x_i - u_{y_i} u_{x_i} \quad i = 1, \ldots, n. \tag{9h}$$

In spite of the fact that reformulation (8) and lower bound (9) are rather simple ones, a branch-and-bound approach based on them turned out to be, by far, the best performing of all the approaches we tested to attack problem (7). However, as we will see through some computational experiments, efficiency of such an approach strongly depends on the use of suitable bound-tightening techniques, which will be discussed in Section 5.1.

4.2 Reformulation based on KKT conditions

Similarly to what already done in [9,23] for box-constrained quadratic programming problems and, very recently, in [7,25], for Standard Quadratic Programming (StQP) problems, one can exploit KKT conditions to reformulate problem (7) as a Mixed Integer Linear Programming (MILP) problem. The KKT conditions for problem (7) are the following

$$\begin{array}{ll}
\mathbf{S}_i\mathbf{x} + \sum_{j=1}^{m}\gamma_j A_{ji} - \lambda - \mu_i = 0 & i = 1,\dots,n \\
\mathbf{x} \in \Delta_n,\ \boldsymbol{\gamma} \in \Delta_m & \\
v \geq \mathbf{A}_j^T\mathbf{x} & j = 1,\dots,m \\
\boldsymbol{\mu} \geq 0 & \\
\mu_i x_i = 0 & i = 1,\dots,n \\
\gamma_j(v - \mathbf{A}_j^T\mathbf{x}) = 0 & j = 1,\dots,m.
\end{array}$$

Let us denote by $X_{\texttt{lin}}$ the region defined by the linear constraints of the KKT conditions (in fact, all of them except the complementarity conditions), and by $X_{\texttt{comp}}$ the region defined by the complementarity conditions. Now let us consider a point $(\mathbf{x}, v, \boldsymbol{\mu}, \boldsymbol{\gamma}, \lambda)$ fulfilling the KKT conditions, i.e., belonging to $X_{\texttt{lin}} \cap X_{\texttt{comp}}$. For each $i \in \{1,\dots,n\}$, let us multiply the first equation by x_i and then sum the result over all i. We end up with

$$\mathbf{x}^T\mathbf{S}\mathbf{x} + \sum_{j=1}^{m}\gamma_j\mathbf{A}_j^T\mathbf{x} - \lambda\sum_{i=1}^{n}x_i - \sum_{i=1}^{n}\mu_i x_i = 0.$$

In view of the complementarity conditions we also have

$$\mathbf{x}^T\mathbf{S}\mathbf{x} + v - \lambda = 0, \tag{10}$$

so that at KKT points the objective function of the problem turns out to be equal to $\frac{1}{2}(\lambda + v)$. Then, since the global minimizer is a KKT point, problem (7) can be rewritten as follows

$$\min_{(\mathbf{x},v,\boldsymbol{\gamma},\boldsymbol{\mu},\lambda)\in X_{\texttt{lin}}\cap X_{\texttt{comp}}} \frac{1}{2}(\lambda + v).$$

In this reformulation the objective function is a linear one and the difficulty of the problem now lies in the complementarity conditions. In order to deal with them, we can introduce the binary variables δ_i, $i = 1,\dots,n$, and ρ_j, $j = 1,\dots,m$, and we will introduce suitable constraints such that

$$\delta_i = \begin{cases} 1 \text{ if } \mu_i = 0 \\ 0 \text{ if } x_i = 0 \end{cases} \qquad \rho_j = \begin{cases} 1 \text{ if } \mathbf{A}_j^T\mathbf{x} = v \\ 0 \text{ if } \gamma_j = 0. \end{cases}$$

In order to introduce suitable constraints we need suitable upper bound values u_{x_i}, u_{μ_i}, u_{γ_j}, $u_{v-\mathbf{A}_j^T\mathbf{x}}$ respectively for x_i, μ_i, γ_j, and $v - \mathbf{A}_j^T\mathbf{x}$. Now, we can reformulate problem (7) as a MILP:

$$\begin{array}{lll}
\min \frac{1}{2}(\lambda + v) & & \\
\quad (\mathbf{x}, v, \boldsymbol{\gamma}, \boldsymbol{\mu}, \lambda) \in X_{\texttt{lin}} & & \\
\quad \mu_i \leq u_{\mu_i}(1 - \delta_i) & i = 1,\dots,n & \\
\quad x_i \leq u_{x_i}\delta_i & i = 1,\dots,n & \\
\quad \gamma_j \leq u_{\gamma_j}\rho_j & j = 1,\dots,m & \\
\quad v - \mathbf{A}_j^T\mathbf{x} \leq u_{v-\mathbf{A}_j^T\mathbf{x}}(1 - \rho_j) & j = 1,\dots,m & \\
\quad \delta_i, \rho_j \in \{0,1\} & i = 1,\dots,n, \quad j = 1,\dots,m. &
\end{array} \tag{11}$$

An obvious upper bound for x_i is $u_{x_i} = 1$. In order to compute an upper bound for μ_i and γ_i we first remark, from (10) and recalling that $\mathbf{S}$ has nonnegative entries, that for $\mathbf{x} \in \Delta_n$

$$\lambda = v + \mathbf{x}^T\mathbf{S}\mathbf{x} \geq v \geq \mathbf{A}_j^T\mathbf{x}, \quad j = 1,\dots,m,$$

so that we can compute upper bounds for μ_i and γ_i by solving the following linear problems

$$\begin{aligned} u_{\mu_i}/u_{\gamma_i} = \max\ & \mu_i/\gamma_i \\ & (\mathbf{x}, v, \boldsymbol{\gamma}, \boldsymbol{\mu}, \lambda) \in X_{\mathtt{lin}} \\ & \lambda \geq \mathbf{A}_j^T\mathbf{x} \qquad\qquad j = 1, \dots, m. \end{aligned}$$

For what concerns an upper bound for $v - \mathbf{A}_j^T\mathbf{x}$, if an upper bound GUB for the optimal value of problem (7) and, thus, also for v, is available, then we can compute it by solving the following linear problem

$$u_{v-\mathbf{A}_j^T\mathbf{x}} = \max_{\mathbf{x}\in\Delta_n} GUB - \mathbf{A}_j^T\mathbf{x}.$$

In fact, all these upper bounds can be strengthened through the bound-tightening techniques described in Section 5.1. The reformulation of problem (7) as a MILP is quite appealing and allows to employ powerful solvers like `CPLEX` or `GUROBI` to solve it. However, while this reformulation turned out to be effective when solving StQP problems (see [7,25]), the computational experiments in Section 6.2 will show that it does not appear to be an effective approach for the solution of problem (7).

4.3 Semidefinite relaxation and convex envelope

Following a standard approach for quadratic problems, problem (7) can be reformulated as follows

$$\begin{aligned} \min\ & \tfrac{1}{2}\mathbf{S} \bullet \mathbf{X} + v \\ & v \geq \mathbf{A}_j^T\mathbf{x} \quad j = 1, \dots, m \\ & \mathbf{x} \in \Delta_n, \\ & \mathbf{X} = \mathbf{x}\mathbf{x}^T. \end{aligned}$$

This immediately leads to the classical semidefinite relaxation for quadratic problems by simply replacing the equality constraint $\mathbf{X} = \mathbf{x}\mathbf{x}^T$ with the semidefinite constraint $\mathbf{X} \succeq \mathbf{x}\mathbf{x}^T$. Unfortunately, the bound obtained by solving the semidefinite relaxation is a trivial one, namely the solution of the linear problem

$$\begin{aligned} \min\ & v \\ & v \geq \mathbf{A}_j^T\mathbf{x}\ j = 1, \dots, m \\ & \mathbf{x} \in \Delta_n. \end{aligned} \tag{12}$$

To see this, it is enough to observe that for each $\mathbf{x} \in \Delta_n$, we can always find a diagonal matrix $\mathbf{X}$ such that $\mathbf{X} \succeq \mathbf{x}\mathbf{x}^T$ holds and, thus, $\mathbf{X}$ is feasible for the semidefinite relaxation. In view of the structure of matrix $\mathbf{S}$, with all diagonal elements equal to 0, it turns out that $\mathbf{S} \bullet \mathbf{X} = 0$ for such diagonal matrix, so that we are left with the simple bound (12).

We end up with the trivial bound (12) even when we replace the quadratic part of the objective function of problem (7) with its convex envelope over Δ_n. Indeed, function $\mathbf{x}^T\mathbf{S}\mathbf{x}$ is edge-concave (see [20,21]), i.e., it is concave over any segment parallel to an edge of the unit simplex. To see this it is enough to show that for all $i, j \in \{1, \dots, n\}$, $i \neq j$, we have that

$$(\mathbf{e}_i - \mathbf{e}_j)^T\mathbf{S}(\mathbf{e}_i - \mathbf{e}_j) = S_{ii} + S_{jj} - 2S_{ij} = -2S_{ij} \leq 0,$$

where $\mathbf{e}_i$ and $\mathbf{e}_j$ are the i-th and j-th vertex of the unit simplex (recall that all diagonal elements of $\mathbf{S}$ are null and all off-diagonal elements are nonnegative). Then, it is well known (see [20,21]) that edge-concave functions have a vertex polyhedral convex envelope, namely, their convex envelope is the affine function interpolating $\mathbf{x}^T\mathbf{S}\mathbf{x}$ at the vertices of the unit simplex Δ_n. Since $\mathbf{S}$ has a null diagonal, this means that the convex envelope is the null function, so that by replacing the quadratic term with its convex envelope over the unit simplex we are back to (12).

5 A branch and bound approach

As already discussed in Section 4.1, given the lower bounds ℓ_{x_i}, ℓ_{y_i} and the upper bounds u_{x_i}, u_{y_i} for variables x_i and y_i respectively, $i \in \{1, \ldots, n\}$, a lower bound for problem (7) over the box $\prod_{i=1}^n [\ell_{x_i}, u_{x_i}] \times \prod_{i=1}^n [\ell_{y_i}, u_{y_i}]$ can be computed by solving the LP problem (9). In the proposed branch-and-bound approach the size of the boxes is progressively reduced through the branching strategy discussed in Section 5.2. However, a reduction merely based on the branching strategy would lead to a quite inefficient algorithm. It turns out that performance can be strongly enhanced by a bound-tightening technique. Such technique, described in the following Section 5.1, is expensive but, as we will see through the computational experiments, extremely rewarding. Next, in Section 5.2 we will briefly discuss the simple branching strategy. Finally, in Section 5.3 we will discuss a technique to strengthen the lower bound, based on the solution of a (small-dimensional) MILP. Note that we do not discuss convergence of the proposed branch-and-bound approach, since it easily follows by rather standard and general arguments which can be found in [10].

5.1 Bound-tightening technique

At each node, associated to a box $\prod_{i=1}^n [\ell_{x_i}, u_{x_i}] \times \prod_{i=1}^n [\ell_{y_i}, u_{y_i}]$, we refine the bounds by recomputing $\ell_{x_i}, \ell_{y_i}, u_{y_i}, u_{x_i}$. All bounds are computed by solving LP problems having the feasible set defined by constraints (9b)-(9h) and the additional constraint

$$\frac{1}{2}\sum_{i=1}^n f_i + v \leq GUB, \tag{13}$$

stating that we are only interested at feasible solutions where the underestimating function, i.e., the left-hand side of the constraint, corresponding to the objective function (9a), is not larger than the current upper bound GUB. In other words, we are applying an optimality-based domain reduction (see, e.g., [5,22]), removing feasible points which do not allow to improve the current best feasible solution. The $4n$ LP problems to be solved have the following objective functions:

1. $\min y_i$ for computing ℓ_{y_i};
2. $\max y_i$ for computing u_{y_i};
3. $\min x_i$ for computing ℓ_{x_i};
4. $\max x_i$ for computing u_{x_i}.

Once we have strengthened the bounds, we recompute the lower bound by solving (9) with the new lower and upper limits for the variables. Note that the underestimating function and, thus, the lower bound depends on such limits. But also the converse is true: the limits depend on the underestimating function. Thus, once we have computed the lower bound, we can solve again the LP problems in order to further reduce the limits. These can be iteratively reduced until the difference between the (non-decreasing) lower bounds at two consecutive iterations fall below a given threshold ϵ (we set $\epsilon = 1.e^{-3}$ in our experiments). Note that rather than solving all $4n$ LP problems with the same feasible set, we could solve each of them with a different feasible region by incorporating all previously computed new limits in the definition of the feasible region for the next limit to be computed. That leads to sharper bounds, however we excluded this opportunity since we observed that LP solvers strongly benefit from the opportunity of solving problems over the same feasible region.
The described bound-tightening technique is quite expensive. In fact, what we observed through our computational experiments is that it is not necessary to solve all $4n$ LPs but it is enough to concentrate the effort on the most 'critical' variables. More precisely, in order to reduce the

number of LPs without compromising the performance, we employed the following strategies. First, we computed the quantities:

$$g_i = x_i^* \mathbf{S}_i \mathbf{x}^* - f_i^*, \tag{14}$$

where $\mathbf{x}^*$ and $\mathbf{f}^*$ are computed at the solution of problem (9). In other words, g_i measures the distance between the real and underestimated value of the bilinear term $x_i y_i$. The larger the distance, the higher is the need for a more accurate underestimation of the bilinear term. Then, we solved the following LP problems.

– $\lceil 0.2n \rceil$ LP problems with objective function min y_i, for all i corresponding to the $\lceil 0.2n \rceil$ largest g_i values;
– a fixed number T ($T = 10$ in our experiments) of LP problems with objective function max y_i, for all i corresponding to the T largest g_i values;
– again T LPs problems with objective function max x_i, for all i corresponding to the T largest g_i values;
– no LP problem with objective function min x_i.

These choices have been driven by some experimental observations. In particular, we noticed that the lower limit for y_i is the most critical for the bound computation or, stated in another way, constraint

$$f_i \geq \ell_{y_i} x_i + \ell_{x_i} y_i - \ell_{x_i} \ell_{y_i},$$

is often the active one. For this reason a larger budget of LP problems is allowed to improve this lower limit with respect to the upper limits. Instead, we never try to improve the lower limit ℓ_{x_i} because it is experimentally observed that this limit can seldom be improved.
This way, the overall number of LPs to be solved at each iteration is reduced to $0.2n + 2T$ (note that the overall number of LPs to be solved at each node can be computed by multiplying this value by the number of iterations, which, however, is not known in advance).
In spite of the reduction of the number of LPs solved, the computational cost per node is still remarkable. But, as we will see in the section about computational experiments, this recursive bound-tightening procedure allows to considerably reduce the number of branch-and-bound nodes.

5.2 Branching Strategy

The branching strategy we employed is a rather standard one. Given a node associated to a box $B_x \times B_y = \prod_{i=1}^n [\ell_{x_i}, u_{x_i}] \times \prod_{i=1}^n [\ell_{y_i}, u_{y_i}]$, and the optimal solution $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{f}^*, v^*)$ of the corresponding relaxation (9), we first select an index $r \in \{1, \dots, n\}$ such that $r \in \arg\max_{i=1,\dots,n} g_i$, where g_i is defined in (14). In other words, we select the index corresponding to the bilinear term where we have the largest error at the optimal solution of the relaxation. Next, we might define the following branching operations:

Branching on x and y: Define four children nodes by adding constraints $\{x_r \leq x_r^*,\ y_r \leq y_r^*\}$, $\{x_r \leq x_r^*,\ y_r \geq y_r^*\}$, $\{x_r \geq x_r^*,\ y_r \leq y_r^*\}$, $\{x_r \geq x_r^*,\ y_r \geq y_r^*\}$, respectively (quaternary branching);
Branching on x: Define two children nodes by adding constraints $x_r \leq x_r^*$ and $x_r \geq x_r^*$, respectively (binary branching);
Branching on y: Define two children nodes by adding constraints $y_r \leq y_r^*$ and $y_r \geq y_r^*$, respectively (binary branching).

Note that all choices above, with the new McCormick relaxation given by the new limits on the variables, reduce to zero the error for bilinear term $x_r y_r$ at the optimal solution $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{f}^*, v^*)$. It is worthwhile to remark that the computed lower bound tends to become exact even when branching is always performed with respect to variables of the same type (say, always variables x_i, $i = 1, \dots, n$). Indeed, only one of the two boxes B_x or B_y needs to converge to a single point in order to let the underestimating function values converge to the original objective function values. This is a consequence of the fact that the McCormick underestimation function tends to the value of the corresponding bilinear term even when only one of the two intervals on which it is defined shrinks to a single point. In the computational experiments we tried all three possibilities discussed above and it turns out that the best choice is the binary branching obtained by always branching on y variables.

5.3 A more expensive bounding strategy

In order to further strengthen the bound, we define a Mixed Integer Linear Programming problem where we select a subset $\mathcal{B}$ of promising variables, again according to the quantity (14), and define a piece-wise McCormick relaxation. In particular, in our experiments we selected the five most promising variables according to (14). Now, let $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{f}^*, v^*)$ be the optimal solution of the final relaxation (9) after bound-tightening. For each $r \in \mathcal{B}$, we define a piece-wise McCormick relaxation (see, e.g., [3]). Namely, we split the current interval $[\ell_{x_r}, u_{x_r}]$ into two sub-intervals $[\ell_{x_r}, x_r^*]$ and $[x_r^*, u_{x_r}]$, and we introduce a binary variable δ_r controlling in which sub-interval we are (the former when $\delta_r = 1$, the latter when $\delta_r = 0$). In a completely analogous way we split the current interval $[\ell_{y_r}, u_{y_r}]$ into two sub-intervals $[\ell_{y_r}, y_r^*]$ and $[y_r^*, u_{y_r}]$, and we introduce a binary variable η_r controlling in which sub-interval we are. Then, we compute the new bound by solving the following MILP problem:

$$\min \frac{1}{2}\sum_{i=1}^{n} f_i + v$$

$$(\mathbf{x}, \mathbf{y}, \mathbf{f}, v) \text{ fulfills (9b)-(9h)}$$

$$\ell_{x_i}\delta_i + x_i^*(1-\delta_i) \le x_i \le x_i^*\delta_i + u_{x_i}(1-\delta_i), \quad i \in \mathcal{B} \tag{15a}$$
$$\ell_{y_i}\eta_i + y_i^*(1-\eta_i) \le y_i \le y_i^*\eta_i + u_{y_i}(1-\eta_i), \quad i \in \mathcal{B} \tag{15b}$$

$$f_i \ge y_i^* x_i + x_i^* y_i - y_i^* x_i^* - u_{x_i}u_{y_i}(2-\delta_i-\eta_i), \quad i \in \mathcal{B} \tag{15c}$$
$$f_i \ge u_{y_i} x_i + x_i^* y_i - u_{y_i} x_i^* - u_{x_i}u_{y_i}(1+\eta_i-\delta_i), \quad i \in \mathcal{B} \tag{15d}$$
$$f_i \ge y_i^* x_i + u_{x_i} y_i - y_i^* u_{x_i} - u_{x_i}u_{y_i}(1+\delta_i-\eta_i), \quad i \in \mathcal{B} \tag{15e}$$
$$f_i \ge u_{y_i} x_i + u_{x_i} y_i - u_{y_i} u_{x_i} - u_{x_i}u_{y_i}(\eta_i+\delta_i), \quad i \in \mathcal{B} \tag{15f}$$
$$f_i \ge y_i^* x_i + x_i^* y_i - y_i^* x_i^* - u_{x_i}u_{y_i}(\delta_i+\eta_i), \quad i \in \mathcal{B} \tag{15g}$$
$$f_i \ge \ell_{y_i} x_i + x_i^* y_i - \ell_{y_i} x_i^* - u_{x_i}u_{y_i}(1+\delta_i-\eta_i), \quad i \in \mathcal{B} \tag{15h}$$
$$f_i \ge y_i^* x_i + \ell_{x_i} y_i - y_i^* \ell_{x_i} - u_{x_i}u_{y_i}(1+\eta_i-\delta_i), \quad i \in \mathcal{B} \tag{15i}$$
$$f_i \ge \ell_{y_i} x_i + \ell_{x_i} y_i - \ell_{y_i} \ell_{x_i} - u_{x_i}u_{y_i}(2-\eta_i-\delta_i), \quad i \in \mathcal{B} \tag{15j}$$

$$\delta_i, \eta_i \in \{0,1\}, \quad i \in \mathcal{B}.$$

Constraints (15a)-(15b) determine the range value of x_i and y_i, respectively, depending on the binary variables δ_i and η_i, whereas constraints (15c)-(15j) establish the McCormick inequalities which need to be taken into account as a function of the binary variables δ_i, η_i or, equivalently, as a function of the sub-interval into which we are. For instance, if $\delta_i = \eta_i = 1$, we have $x_i \in [\ell_{x_i}, x_i^*]$ and $y_i \in [\ell_{y_i}, y_i^*]$, so that the McCormick inequalities which need to be considered are (15c) and (15j), while all the others become redundant constraints (note that $u_{x_i}u_{y_i}$ is an upper bound for the bilinear term $x_i y_i$ over the current intervals for x_i and y_i). We remark that the optimal value of the MILP is the lowest possible lower bound among all nodes obtained through a sequence of quaternary branching operations performed over the variables in $\mathcal{B}$ with branching points (x_i^*, y_i^*), $i \in \mathcal{B}$, if no bound-tightening is performed at these nodes.

If the cardinality of $\mathcal{B}$ is small (five in our experiments) the time needed to compute this bound at a given node of the branch-and-bound tree is small compared to the time needed for the bound-tightening operation, so that we can improve the quality of the bound without significatively increasing the computational cost per node. Note that the intervals $[\ell_{x_i}, u_{x_i}]$ and $[\ell_{y_i}, u_{y_i}]$ could be split into more than two sub-intervals with the addition of suitable binary variables. However, our computational experiments suggest that this does not lead to significative improvements.

6 Numerical Results

6.1 Test instances description

The game is about spot checking a set of n places to guard them against an adversary. The places are spatially scattered, with a directed weighted graph describing the connections (direct reachability) of place v from place u by an edge $v \to u$ with a random length.

The payoffs in the game are given by an $n \times n$-Matrix $\mathbf{A}$ (so $m = n$ in the above description), and are interpreted as the loss that the defending player 1 suffers when checking place i while the attacker is at place j. Thus, the defender can:

- either miss the attacker ($i \neq j$) in which case there will be a Weibull-distributed random loss with shape parameter 5 and scale parameter 10.63 (so that the variance is 5);
- or hit the attacker at $i = j$, in which case there is zero loss.

The defender is thus minimizing, and the attacker is maximizing. The problem above is that of the defender. The Nash equilibrium then gives the optimal random choice of spot checks to minimize the average loss. To avoid trivialities, the payoff matrices are constructed not to admit pure strategy equilibria, so that the optimum (without switching cost) is necessarily a mixed strategy.

As for the switching cost, if the defender is currently at position i and next – according to the optimal random choice – needs to check the (non-adjacent) place j, then the cost for the switch from i to j is the shortest path in the aforementioned graph (note that, since the graph is directed, the matrix $\mathbf{S}$ is generally nonsymmetric).

For the random instances, the matrix $\mathbf{S}$ is thus obtained from a (conventional) all-shortest path algorithm applied to the graph. Note that the graph is an Erdös-Renyi type graph with n nodes and $p = 0.3$ chance of any two nodes having a connection.

The weights in the graph were chosen exponentially distributed with rate parameter $\lambda = 0.2$, and the Weibull distribution for the losses has a shape parameter 5 and scale parameter ~ 10.63, so that both distributions have the same variance of 5.

The graph sizes considered are $n = 50, 75, 100$ and for each graph size we consider ten random instances.

We shift the matrix $\mathbf{S}$ to make less relevant the quadratic part that is more difficult to bound. More in detail, given the vector $\mathbf{c} \in \mathbb{R}^n$ with

$$c_i = \min_{j \neq i} \{S_{ij}\},$$

we have on the feasible set (i.e., exploiting $\mathbf{e}^T\mathbf{x} = 1$) that

$$\mathbf{x}^T\mathbf{S}\mathbf{x} = \mathbf{x}^T\bar{\mathbf{S}}\mathbf{x} + \mathbf{c}^T\mathbf{x},$$

where

$$\bar{\mathbf{S}} = \mathbf{S} - \begin{pmatrix} c_1\mathbf{e}^T \\ \vdots \\ c_n\mathbf{e}^T \end{pmatrix}$$

Then our objective function in the practical implementation becomes

$$\frac{1}{2}(1-\alpha)\mathbf{x}^T\bar{\mathbf{S}}\mathbf{x} + \frac{1}{2}(1-\alpha)\mathbf{c}^T\mathbf{x} + \alpha \max_{i=1,\ldots,m} \mathbf{A}_i^T\mathbf{x},$$

and we tested α values in $\{0.3, 0.4, \ldots, 0.9\}$.

Note that all the data of the test instances are available at the web site `http://www.iasi.cnr.it/~liuzzi/StQP`.

6.2 A comparison with the existing QP literature

The problem discussed in this paper belongs to the class of nonconvex QP problems with linear constraints, which is a quite active research area. Even well-known commercial solvers, like `CPLEX` and `GUROBI`, have recently included the opportunity of solving these problems. In [25] different solution approaches have been compared over different nonconvex QP problems, namely: Standard Quadratic Programming problems (StQP), where the feasible region is the unit simplex; BoxQP, where the feasible region is a box; and general QPs, where the feasible region is a general polytope. The tested approaches have been the nonconvex QP solver of `CPLEX`, `quadprogBB` (see [6]), `BARON` (see [17]), and `quadprogIP`, introduced in [25] itself. According to the computational results reported in that work, depending on the class of QP problems considered, the best performing approaches are the nonconvex QP solver of `CPLEX` and `quadprogIP`. For this reason we ran these solvers over our test instances, with the addition of the nonconvex QP solver of `GUROBI`.

For these experiments we employed an Intel® Core i7-8550U CPU at 1.8GHz with 4 cores and 16GB main memory. Since even at dimension $n = 50$ and for different α values most instances were not solved within a time limit of 3 hours, we do not report extensive results. Rather, we only discuss the behavior over a single instance with $n = 50$ and $\alpha = 0.5$, which, however, is representative of the behavior over the other instances. We first ran the nonconvex QP solvers both of `CPLEX` and of `GUROBI`. Both were unable to terminate within 3 hours. The relative percentage gap still to be closed after that time was 3.61% and 5.63% for `CPLEX` and `GUROBI`, respectively.

Next, we ran `quadprogIP`. This is based on a MILP reformulation of nonconvex QP problems with linear constraints, which in some cases appears to be a rather promising way to tackle them. Since this solver requires lower and upper bounds for all variables, we needed to introduce a lower and upper bound for variable v. Due to nonnegativity of vectors $\mathbf{A}_j$, $j = 1, \ldots, m$, a simple lower bound for v is 0, while a valid upper bound, fulfilled by optimal solutions of the problem, is

$$\max_{j=1,\ldots,m,\ i=1,\ldots,n} A_{ji}.$$

After 3 hours the percentage gap still to be closed was 17.88%. However, we made a further experiment. After observing that a critical aspect of `quadprogIP` is the computation of valid upper bounds for primal and dual variables, we introduced a pre-processing phase, where all the upper bounds u_{x_i}, u_{μ_i}, u_{γ_j}, $u_{v-\mathbf{A}_j^T\mathbf{x}}$, respectively for x_i, μ_i, γ_j, and $v - \mathbf{A}_j^T\mathbf{x}$, appearing in the MILP reformulation (11) have been strengthened via the optimality-based bound-tightening technique discussed in Section 5.1 (an estimate GUB for the globally optimal value was obtained by running some local searches from randomly generated points). After that we ran `GUROBI` and `CPLEX` to solve the MILP reformulation (11). The pre-processing phase definitely helped, but still none of the two solvers was able to terminate within the 3 hours time limit. In this case `GUROBI` performed better and terminated after 3 hours with a percentage gap of 0.25 % still to be closed, while `CPLEX` terminated after the same time with a percentage gap of 1.66 % still to be closed.

Finally, we ran the approach proposed in this paper and it terminated after 114 seconds returning an optimal solution with a relative tolerance value $\varepsilon = 10^{-6}$. The main reason for the much better performance of our approach lies in the optimality-based bound-tightening technique. To confirm this fact, we made a further experiment by running the same approach but without activating the bound-tightening procedure. After 3 hours the gap still to be closed was equal to 4.73%.

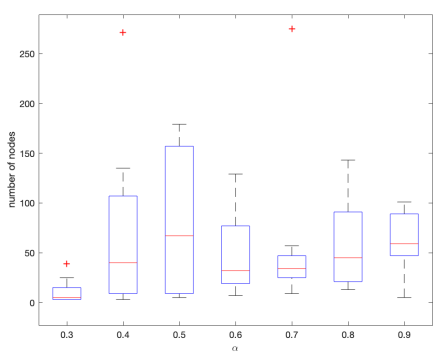
In the following subsection we report the results attained by our approach for different n and α values and we comment the results.

6.3 Computational results over the proposed test instances

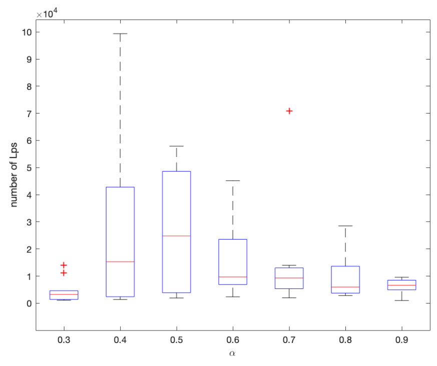
In this section we show through some computational experiments that our approach successfully solves the problem for the sizes ($50 \leq n \leq 100$) of the real instances arising from the game theory application discussed in Section 2. Moreover, we show it scales well with the dimension, outperforming all the available software for solving non convex QPs, as already observed in the previous subsection.

The algorithm has been coded using the Julia [4] language (version 1.3.1). Doing the implementation we parallelized as much as possible the bound-tightening procedure discussed in Section 5.1, where many LPs with the same feasible region need to be solved. The code is available for download at the URL `http://www.iasi.cnr.it/~liuzzi/StQP`. All the experiments have been carried out on an Intel® Xeon® gold 6136 CPU at 3GHz with 48 cores and 256GB main memory. This machine is more powerful with respect to the one employed in the experiments discussed in the previous subsection. Just to give an idea of the relative performance, we remark that with the machine used for the previous experiments, the largest time to solve instances with $n = 50$ is approximately 600 seconds, while with the machine employed for the experiments in this section the largest time is approximately 120 seconds.

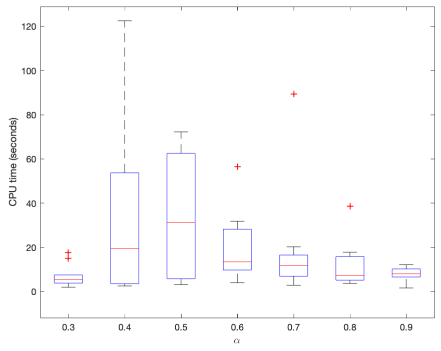
We have solved ten instances for three different sizes $n = 50,\ 75,\ 100$ and for the following values of $\alpha = \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Note that lower and larger values of α with respect to those we tested give rise to much simpler instances (recall that the problem becomes polynomially solvable for the extreme values $\alpha = 0$ and $\alpha = 1$). We set a time limit of 10800s for all instances. Differently from the previous subsection, where we stopped when a relative tolerance $\varepsilon = 10^{-6}$ was used to comply with the one of the existing solvers, in this section we fixed a relative tolerance $\varepsilon = 10^{-3}$, which is considered adequate for practical applications. For $n = 50$ and $n = 75$ we solve all the instances to optimality (in fact, the largest time to solve an instance with $n = 50$ is about 2 minutes, whereas for $n = 75$ the largest time is below 1 hour, but most of the problems are solved within 10 minutes). In Figures 2a-2c we report the box plot of number of nodes, number of LPs and CPU time needed for the different values of α when $n = 50$. The figure shows that the hardest instances are the ones corresponding to the central values of α and this will turn out to hold true also at larger dimensions. We also observe that the overall number of nodes is extremely limited, thus confirming that, while computationally expensive, the bound-tightening procedure allows to considerably reduce the size of the branch-and-bound tree (again, this fact is observed also at larger dimensions). We also stress that $n = 50$ is a typical size for real instances of this problem arising from the game theory application discussed in Section 2. In Figures 3, we report the different box plots for all the instances at $n = 75$. It is worthwhile to remark that we solve most of them within ten minutes and exploring less than 300 nodes. Finally, in Figure 4 we report the performance of our method on problems of dimension $n = 100$. In this case there are seven instances we are not able to solve within the time limit. These occur for values $\alpha \in \{0.6, 0.7, 0.8\}$, thus confirming that the central values of this parameter give rise to the most challenging instances. With respect to $n = 50$ and $n = 75$, we have the additional box plot displayed in Figure 4d reporting the final percentage gap when the time limit is reached. Note that it is never larger than 1.2% and most of the times it is lower than 0.5%, thus showing that, even when the algorithm does not terminate, the quality of the returned solution is guaranteed to be high. We also tried to apply the bound strategy described in Section 5.3 on the seven instances that we did not manage to solve to optimality. The result are shown in Table 1, and it can be observed that the MILP-based



(a) Total number of nodes for $n = 50$



(b) Number of LPs solved for $n = 50$



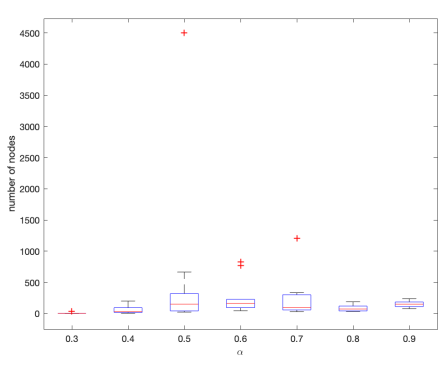
(c) CPU times in seconds

Fig. 2: Box plots for different performance measures for $n = 50$

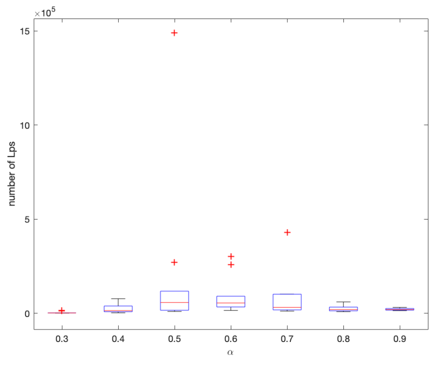
approach manages to reduce the gap within the three hours, but does not improve the solution enough to close the gap. In general, when the bound tightening approach is used, the gap decreases faster, but the price to pay at each node is clearly higher, and the gain is not significant enough to close the gap a lot faster. It may be worth to use this strategy when one has a limited time budget since it allows to produce a better quality solution. But, on the other hand, it appears that only a limited number of nodes which are not closed by the bound-tightening procedure can instead be closed by the MILP-based bound. This is remarkable since, as previously commented in Section 5.3, the MILP-based bound, in fact, corresponds to the lowest bound obtained after five consecutive quaternary branching operations (without bound-tightening), i.e., after the generation of 1024 nodes.

Problem		LP based			MILP based			
α	#	# nodes	# numLP	% gap	# nodes	# numLP	# numMILP	% gap
0.6	10	13003	3977151	5.11E-03	9921	2896305	9785	4.27E-03
0.6	35	7321	3668797	3.41E-03	6445	3310264	3860	2.84E-03
0.6	45	8381	3569883	3.92E-03	7405	3083398	6074	3.10E-03
0.7	10	11417	4054621	4.70E-03	9119	3292767	6637	3.47E-03
0.7	20	13001	4354802	1.17E-02	9487	3056131	9403	1.02E-02
0.7	45	13409	4529598	1.05E-02	9071	3050473	8972	8.92E-03
0.8	20	12433	4544056	1.16E-02	9197	3273988	7529	9.19E-03

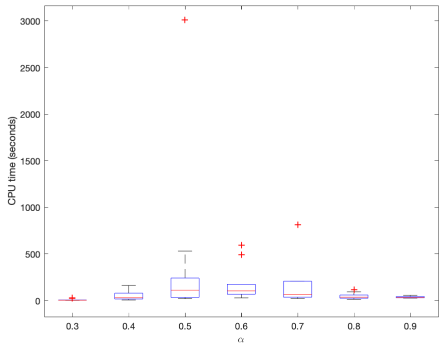
Table 1: Bound tightening results on the instances not solved to optimality



(a) Total number of nodes for $n = 75$



(b) Number of LPs solved for $n = 75$

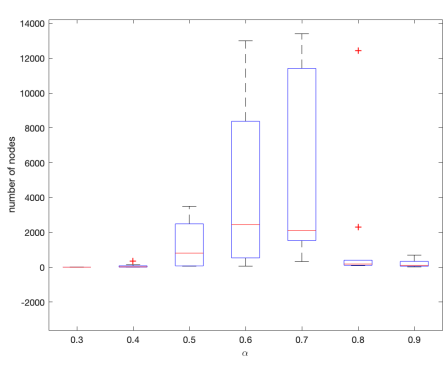


(c) CPU times in seconds

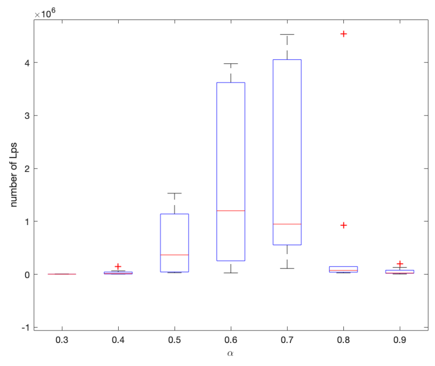
Fig. 3: Box plots for different performance measures for $n = 75$

References

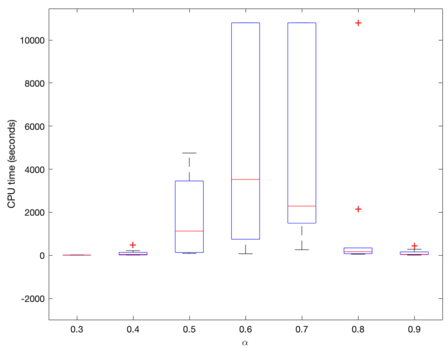
1. Alpcan, T., Başar, T.: Network security: A decision and game-theoretic approach. Cambridge University Press (2010)
2. Alpern, S., Morton, A., Papadaki, K.: Patrolling games. Operations research **59**(5), 1246–1257 (2011)
3. Bergamini, M.L., Aguirre, P., Grossmann, I.: Logic-based outer approximation for globally optimal synthesis of process networks. Computers & chemical engineering **29**(9), 1914–1933 (2005)
4. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM review **59**(1), 65–98 (2017). URL `https://doi.org/10.1137/141000671`
5. Caprara, A., Locatelli, M.: Global optimization problems and domain reduction strategies. Mathematical Programming **125**(1), 123–137 (2010)
6. Chen, J., Burer, S.: Globally solving nonconvex quadratic programming problems via completely positive programming. Mathematical Programming Computation **4**(1), 33–52 (2012)
7. Gondzio, J., Yildirim, E.A.: Global solutions of nonconvex standard quadratic programs via mixed integer linear programming reformulations. arXiv preprint arXiv:1810.02307 (2018)
8. Hansen, K.A., Koucky, M., Lauritzen, N., Miltersen, P.B., Tsigaridas, E.P.: Exact algorithms for solving stochastic games. In: Proceedings of the forty-third annual ACM symposium on Theory of computing, pp. 205–214 (2011)
9. Hansen, P., Jaumard, B., Ruiz, M., Xiong, J.: Global minimization of indefinite quadratic functions subject to box constraints. Naval Research Logistics (NRL) **40**(3), 373–392 (1993)
10. Horst, R., Tuy, H.: Global optimization: Deterministic approaches (2nd edition). Springer Science & Business Media (2013)
11. Lozovanu, D., Solomon, D., Zelikovsky, A.: Multiobjective games and determining pareto-nash equilibria. Buletinul Academiei de Ştiinţe a Republicii Moldova. Matematica **3**, 115–122 (2005)
12. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part iconvex underestimating problems. Mathematical programming **10**(1), 147–175 (1976)
13. Rass, S., Alshawish, A., Abid, M.A., Schauer, S., Zhu, Q., De Meer, H.: Physical intrusion gamesoptimizing surveillance by simulation and game theory. IEEE Access **5**, 8394–8407 (2017)
14. Rass, S., König, S.: Password security as a game of entropies. Entropy **20**(5), 312 (2018)



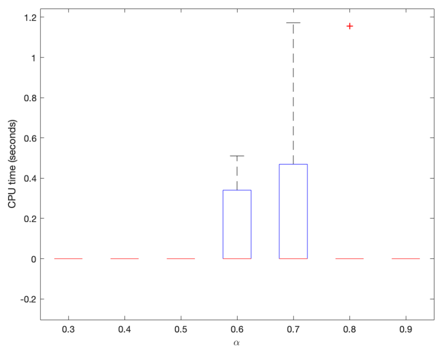
(a) Total number of nodes for $n = 100$



(b) Number of LPs solved for $n = 100$



(c) CPU times in seconds



(d) % Gap at the time limit

Fig. 4: Box plots for different performance measures for $n = 100$

15. Rass, S., König, S., Schauer, S.: On the cost of game playing: How to control the expenses in mixed strategies. In: International Conference on Decision and Game Theory for Security, pp. 494–505. Springer (2017)
16. Rass, S., Rainer, B.: Numerical computation of multi-goal security strategies. In: International Conference on Decision and Game Theory for Security, pp. 118–133. Springer (2014)
17. Sahinidis, N.V.: Baron: A general purpose global optimization software package. Journal of global optimization **8**(2), 201–205 (1996)
18. Shapley, L.S.: Stochastic games. Proceedings of the national academy of sciences **39**(10), 1095–1100 (1953)
19. Tambe, M.: Security and game theory: algorithms, deployed systems, lessons learned. Cambridge university press (2011)
20. Tardella, F.: On the existence of polyhedral convex envelopes. In: Frontiers in global optimization, pp. 563–573. Springer (2004)
21. Tardella, F.: Existence and sum decomposition of vertex polyhedral convex envelopes. Optimization Letters **2**(3), 363–375 (2008)
22. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. Mathematical programming **99**(3), 563–591 (2004)
23. Vandenbussche, D., Nemhauser, G.L.: A polyhedral study of nonconvex quadratic programs with box constraints. Mathematical Programming **102**(3), 531–557 (2005)
24. Wachter, J., Rass, S., König, S.: Security from the adversarys inertia–controlling convergence speed when playing mixed strategy equilibria. Games **9**(3), 59 (2018)
25. Xia, W., Vera, J.C., Zuluaga, L.F.: Globally solving nonconvex quadratic programs via linear integer programming techniques. INFORMS Journal on Computing (2019)