INFORMATICA, 2009, Vol. 20, No. 1, 79–98

A Web-Based Bimatrix Game Optimization Model of Polynomial Complexity

Jonas MOCKUS
Institute of Mathematics and Informatics
Akademijos 4, LT-08663 Vilnius, Lithuania
e-mail: jmockus@gmail.com

Received: February 2008; accepted: April 2008

Abstract. The objective of this paper is the description, justification, and web-based implementation of polynomial time algorithms for equilibrium search of Quadratic Bimatrix Games (QBG). An algorithm is proposed combining exact and heuristic parts. The exact part has the Irelevant Fraud (IF) component for cases when an equilibrium exists with no pure strategies. The Direct Search (DS) component finds a solution if an equilibrium exists in pure strategies. The heuristic Quadratic Strategy Elimination (QSE) part applies IF and DS to reduced matrices obtained by sequential elimination of strategies that lead to non-positive IF solutions. Finally, penalties needed to prevent unauthorized deals are calculated based on Nash axioms of two-person bargaining theory. In the numeric experiments QSE provided correct solution in all examples. The novel results include necessary and sufficient conditions when the QBG problem is solved by IF algorithm, the development of software and the experimental testing of large scale QBG problems up to $n = 800$. The web-site `http://pilis.if.ktu.lt/~jmockus` includes this and accompanying optimization models.
Keywords: game theory, heuristics, optimization, economics, education.

Introduction

Traditional economical models of competition define prices and other production parameters that satisfy the Nash equilibrium (Nash, 1950). However, the market competition represents just a part of competitive economical and social activities. Inspection is another important part of such activities. Objectives of inspection include tax collection, property, health, and environment protection.

This paper investigates an example of quadratic bimatrix game model that describes the competition between inspectors and violators. We call them Inspector Games (IG). The IG is similar but not identical to the well known Inspection Game (Oven, 1995). The objective is to illustrate how the theories of games (Oven, 1995; Forgo *et al.*, 1999; McKelvey and McLennan, 1996; McKelvey *et al.*, 2005; Games, 2006) and optimization (Dantzig, 1963; Karmarkar, 1984; Murty, 2006) can be applied to inspection problems defined as Quadratic Bimatrix Game (QBG), to show the limitations imposed by the high complexity of the problem, and investigate ways to address them.

To solve large scale game problems and to prepare examples of game theory studies, it is essential to use polynomial time algorithms. No polynomial time algorithm is known for obtaining Nash Equilibrium (NE) of bimatrix games in general (Porter *et al.*, to appear). Therefore, an important task is to define a subset BG problems where NE can be obtained in polynomial time. In the paper this is done for QBG if NE exists in strictly mixed strategies (no pure strategies) or if NE exists in pure strategies. For other QBG a polynomial time heuristic is applied.

In (Forgo *et al.*, 1999; Berg, 2000) necessary and sufficient conditions of NE are given for Bimatrix Games (BG) in terms of a bilinear programming problem. No polynomial time algorithms are known for this problem. In the paper a simple proof is provided that the Irrelevant Fraud (IF) model described in (Mockus, 1997) defines the sufficient and necessary conditions for QBG if NE exists with no pure strategies. The IF model uses Linear Programming (LP) for solution. If NE exists in pure strategies then NE can be found by the Direct Search (DS) of pure strategies.

To solve IG problems in general the heuristic algorithm is proposed, implemented, and investigated. This algorithm, for short QSE, uses strategy elimination and includes both IF and DS. The heuristic part makes NE search by sequentially eliminating strategies that lead to infeasible IF solutions. This is a heuristic implementation of the general idea of strategy elimination (Knuth *et al.*, 1988; Conitzer and Sandholm, 2005a; Conitzer and Sandholm, 2005b). In the numeric experiments with matrix dimensions up to 800 the QSE heuristic provided correct solution for all IG. Automatic testing for optimality included in QSE ensured that only correct solutions were accepted. The CPU time ranged from 25 minutes for the largest dimensions, to 20 seconds for $n = 200$.

The game model of this paper is implemented as a Java applet in a system used for graduate studies and scientific collaboration in the field of optimization and market games (Mockus, 2007; Mockus, 2003b; Mockus, 2006b; Mockus, 2006a; Heilo and Mockus, 2008; Mockus, 2003a). The system includes theory and software of different optimization models in 85,000 files. The main web-site: `http://soften.ktu.lt/~mockus`.

Published examples of this system is optimization of stock exchange model (Mockus, 2003b) and Walras competition model (Mockus, 2004).

The opportunity to run software developed by colleagues in a Web browser facilitates for scientific collaboration and distance learning. Results obtained by other researchers can be easily tested by running their software with our input. Therefore algorithms, software and results published in the scientific papers can be easily replicated and independently investigated. So far, this possibility has not been widely adopted. We include the snapshots of graphical user interfaces to simplify testing of the results and to illustrate how the calculations were performed by authors.

To provide this experience in the Web browser environment we need a platform independent language running software on remote computers, for example, Java, Perl, Python, PHP. We chose Java because we felt it is more efficient than other alternatives for large scale optimization problems.

1. Bimatrix Game (BG)

1.1. *Complexity*

The well known results of algorithm complexity show the limitations of exact solutions. That explains popularity of heuristic algorithms.

An important open problem in complexity theory is the existence of polynomial-time algorithms for computing equilibrium for subsets of bimatrix games. For games in extensive form, the reduced normal form may be exponentially large (von Stengel, 1998).

The Lemke-Howson algorithm (Forgo *et al.*, 1999) is the classical algorithm for the problem of finding a Nash equilibrium of a bimatrix game. It solves a linear complementary problem and provides a constructive and elementary proof of existence of an equilibrium.

The paper (Savani and von Stengel, 2004) presents a class of bimatrix games for which the Lemke-Howson algorithm takes, even in the best case, exponential time in the dimension of the game.

The book (Murty, 2006) shows that the computational effort required to solve a linear complementarity problem, by either of the two well known complementary pivot methods is not bounded above by a polynomial in the size of the problem. In was shown that to solve the problem of order n, either of the two methods goes through 2^n pivot steps before termination.

In this paper a polynomial time heuristic QSE is investigated. The IE component of QSE provides exact solution if NE exists with no pure strategies. The DS component solves the problem if NE exists in pure strategies. The QSE heuristic did reach NE in all randomly generated IG during large scale experimentation up to dimension $n = 800$. The QSE heuristic was designed for IG but it can be used for any QBG.

1.2. *Profit Functions*

The payoff of the first player is expressed by a matrix $u(i, j)$ where $i = 1, \ldots, m$ denote moves (pure strategies) of the first player and $j = 1, \ldots, n$ represent moves of the second. The payoff of the second player is $v(i, j)$. The profit functions U and V of the first and second players are defined as expected values of payoffs $u(i, j)$ and $v(i, j)$

$$U(x, y) = \sum_{i,j} x_i u(i, j) y_j, \tag{1}$$

and

$$V(x, y) = \sum_{i,j} x_i v(i, j) y_j. \tag{2}$$

Here $x = (x_i,\ i = 1, \ldots, m)$ and $y = (y_j,\ j = 1, \ldots, n)$ are probabilities (mixed strategies) of moves i and j.

In the Matrix Game (MG) $u(i,j) = -v(ij)$. In the Bimatrix Game (BG) $u(i,j) \neq -v(i,j)$. In the Quadratic Bimatrix Game (QBG) $m = n$.

Often the payoffs are defined as functions of parameters. For example, in the Inspector Game (IG) the payoffs (33) and (34) are defined by three main parameters: the probability p_i to detect a violation if it happens in the area i, the probability q_i of the violation, and the payoff g_i of the violation.

Two "nti-corruption" parameters defining the minimal expected penalty (49) can be regarded, too. The penalty ω for unauthorized deal and a probability p_ω of the penalty ω.

2. Search for Equilibrium of QBG

2.1. *Necessary and Sufficient Conditions*

The necessary and sufficient conditions of NE for BG in general are well known and described in (Forgo *et al.*, 1999; Berg, 2000) as a bilinear programming problem that can be rewritten as a linear complementary problem. No polynomial time algorithms are known for these problems (Murty, 2006). If NE of QBG exists in strictly mixed strategies then the necessary and sufficient conditions are formulated as a system of linear inequalities that can be conveniently solved by efficient polynomial time algorithms of Linear Programming (LP) such as (Karmarkar, 1984; Murty, 2006). Here is a simple proof to explain these conditions.

DEFINITION 1.
A game strategy x: $x_i \geqslant 0$, $\sum_{i=1}^m x_i = 1$ is called the mixed strategy.
A game strategy x: $x_i > 0$, $\sum_{i=1}^m x_i = 1$ is called the strictly mixed strategy.
A game strategy x: $x_i = \{0,1\}$, $\sum_{i=1}^m x_i = 1$ is called the pure strategy.

Theorem 1. *For a pair*

$$x^* > 0, \quad y^* > 0. \tag{3}$$

to be a Nash equilibrium in strictly mixed strategies of the quadratic bimatrix game it is necessary and sufficient that there exist real numbers (α^*, β^*) *such that* $(\alpha^*, \beta^*, x^*, y^*)$ *satisfies the system*

$$Uy^* = \alpha^* I, \tag{4}$$

$$x^* V = \beta^* I, \tag{5}$$

$$Ix^* = 1, \quad Iy^* = 1. \tag{6}$$

where I *denotes the unit vector.*

Proof. Applying the well-known (Forgo *et al.*, 1999) necessary and sufficient conditions of Nash equilibrium for general bimatrix games (U, V) to a subset of bimatrix games in

strictly mixed strategies we can write

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_i u_{i,j} y_j = \alpha, \tag{7}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_i v_{i,j} y_j = \beta, \tag{8}$$

$$\sum_{j=1}^{n} u_{i,j} y_j \leqslant \alpha, \quad i = 1, \ldots, m, \tag{9}$$

$$\sum_{i=1}^{m} x_i vi, j \leqslant \beta, \quad j = 1, \ldots, n, \tag{10}$$

$$\sum_{i=1}^{m} x_i = 1, \quad \sum_{j=1}^{n} y_j = 1, \tag{11}$$

$$x_i > 0, \quad y_j > 0. \tag{12}$$

The difference from the conditions (Forgo *et al.*, 1999) are strict inequalities (12).

From (4), (5)

$$\sum_{j=1}^{m} u_{i,j} y_j = \alpha, \quad i = 1, \ldots, m, \tag{13}$$

$$\sum_{i=1}^{m} x_i v_{i,j} = \beta, \quad j = 1, \ldots, m. \tag{14}$$

From (13), (14), (7), (8)

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_i u_{i,j} y_j = \sum_{i=1}^{m} x_i \sum_{j=1}^{n} u_{i,j} y_j = \sum_{i=1}^{n} x_i \, \alpha = \alpha \sum_{i=1}^{n} x_i = \alpha, \tag{15}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_i v_{i,j} y_j = \sum_{j=1}^{m} y_j \sum_{j=1}^{n} x_i v_{i,j} = \sum_{j=1}^{n} y_j \, \beta = \beta \sum_{j=1}^{n} y_j = \beta. \tag{16}$$

That proves the sufficiency of (4), (5), (6), (3).

To prove the necessity of (4), (5), (6), (3) assume that

$$\sum_{j=1}^{m} u_{1,j} y_j = \alpha - \epsilon, \quad \epsilon > 0, \tag{17}$$

$$\sum_{j=1}^{m} u_{i,j} y_j = \alpha, \quad i = 2, \ldots, m, \tag{18}$$

Then

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_i u_{i,j} y_j = (\alpha-\epsilon)x_1 + \alpha\sum_{i=2}^{m} x_i = \alpha - \epsilon x_1 < \alpha. \tag{19}$$

That proves the necessity.

2.2. *Direct Testing*

The equilibrium can be tested directly by the conditions of Nash equilibrium. First define the "contract" profits U^*, V^* assuming that both players keep the contract solution (x^*, y^*)

$$U^* = \sum_{i,j} x_i^* u(i,j) y_j^*, \tag{20}$$

$$V^* = \sum_{i,j} x_i^* v(i,j) y_j^*. \tag{21}$$

Then define the maximal profits $U_{\max}, V_{\max}$ of players assuming that opposite players keep the contract solution (x^*, y^*)

$$U_{\max} = \max_x \sum_{i,j} x_i u(i,j) y_j^*, \tag{22}$$

$$V_{\max} = \max_y \sum_{i,j} x_i^* v(i,j) y_j. \tag{23}$$

The contract profits U^*, V^* are compared with the values $U_{\max}, V_{\max}$, and if

$$U^* \geqslant U_{\max}, \quad V^* \geqslant V_{\max}. \tag{24}$$

the contract solution $x_i^* > 0$, $y_j^* > 0$ is recorded as an equilibrium strategy.

2.3. *Irrelevant Fraud Model (IF)*

The Irrelevant Fraud model defines conditions where no incentive to break the contract solution (x^*, y^*) exists. Expressions (13), (14), (6) is a system of $2m + 2$ linear equations that can be solved by standard algorithms of linear algebra. However the Linear Programming (LP) is convenient for analysis.

In LP all the variables should be non-negative. Thus we express all the original variables as differences of two non-negative LP variables

$$\begin{aligned}
&U = u_1 - u_2, \quad V = v_1 - v_2, \;\; u_1 \geqslant 0, \;\; u_2 \geqslant 0, \;\; v_1 \geqslant 0, \;\; v_2 \geqslant 0,\\
&y_j = y_j^1 - y_j^2, \quad y_j^1 \geqslant 0, \;\; y_j^2 \geqslant 0, \;\; j = 1, \ldots, m,\\
&x_i = x_i^1 - x_i^2, \quad x_i^1 \geqslant 0, \;\; x_i^2 \geqslant 0, \;\; i = 1, \ldots, m.
\end{aligned} \tag{25}$$

Then from expressions (13), (14), (6) follows this LP problem

$$\max_{x,y,u,v}(u_1 - u_2 + v_1 - v_2), \tag{26}$$

$$\sum_{j=1}^{m} u(i,j)y_j = u_1 - u_2, \quad i = 1,\ldots,m, \tag{27}$$

$$\sum_{i=1}^{m} v(i,j)x_i = v_1 - v_2, \quad j = 1,\ldots,m, \tag{28}$$

$$\sum_{j=1}^{m} y_j = 1, \tag{29}$$

$$\sum_{i=1}^{m} x_i = 1, \tag{30}$$

where $x = (x_1,\ldots,x_m)$, $y = (y_1,\ldots,y_m)$, $u = (u_1,u_2)$, $v = (v_1,v_2)$. If the LP solution happens to be positive, meaning that $x_i > 0$. $y_j > 0$, $i,j = 1,\ldots,m$, than according to the conditions of Theorem 1 that is NE. Variables x_i and y_j can be zero or negative, too. That provides some additional information about the problem when no positive solution exists.

Theorem 2. *If exists, the Nash equilibrium in strictly mixed strategies of quadratic bimatrix game can be defined by an algorithm of polynomial complexity.*

Proof. If positive, the solution of IF model by LP algorithm (26)–(30) satisfies conditions (4)–(3) of Theorem 1. Suppose that the Interior Point algorithm is used solving the LP problem (26)–(30). This algorithm is of polynomial complexity (Murty, 2006; Karmarkar, 1984). That proves the theorem.

That is a theoretical result. In large scale practical problems the simplex algorithm of LP problems can be more efficient since its average complexity is low. The simplex algorithm of LP is of exponential complexity in the worst, and of low degree polynomial complexity in average (Smale, 1983).

Thus, if exists, the positive IF solution defines the Nash equilibrium for the quadratic bimatrix game (1)(2). The problem is that positive solutions of IF algorithm not always exists. An example is the Nash equilibrium in pure strategies. The Direct Search (DS) can be applied for a search of equilibrium in pure strategies (32). If that fails, too, then the heuristic Quadratic Strategy Elimination (QSE) algorithm (Section 2.5) is used.

2.4. *Direct Search (DS)*

It is well known (Forgo *et al.*, 1999) that, in large randomly generated bimatrix $n \times n$ games, the probability that a game has exactly k pure strategy equilibrium is defined by

the asymptotic distribution

$$\lim_{n\to\infty} P_k(n) = \mathrm{e}^{-1}/(k!). \tag{31}$$

Summing the probability over $k = 1, \ldots, n^2$ we find that roughly two-thirds ($1 - \mathrm{e}^{-1}$ as $n \to \infty$) of randomly generated bimatrix games have an equilibrium point in pure strategies. Additional information about asymptotic properties of randomly generated bimatrix games is in (McLennan and Berg, 2005).

Denote by $I(j)$ a set of pairs of indexes $I(j) = \cup_j(i(j), j)$, where index $i(j) = arg\max_i u(i,j)$ defines the maximal element of column j of the inspector matrix $u(i,j)$. Denote by $J(i)$ a set of pairs of indexes $J(i) = \cup_i(i, j(i))$, where index $j(i) = arg\max_j v(i,j)$ defines the maximal element of row i of violator matrix $v(i,j)$. Denote unions of these sets $I = \cup_j I(j)$ and $J = \cup_i J(i)$. Intersection of these unions

$$IJ = I \cap J, \tag{32}$$

defines NE in pure strategies. Empty intersection means that there exists no equilibrium in pure strategies. Defining intersection and defining maximal elements are simple operations requiring n^2 comparisons, therefore DS is a polynomial time algorithm.

2.5. *Quadratic Strategy Elimination Algorithm* (*QSE*)

The general idea of QSE is to eliminate strategies that lead to non-positive IF solutions. First an IF solution for both players x, y is generated. If for some k $x_k \leqslant 0$ or $y_k \leqslant 0$ and the DS solution is not found, then both the strategies x_k and y_k are eliminated from the set of feasible strategies by setting them to zero $x_k = y_k = 0$.

Iterative application of this method generates a sequence of QBG. The iteration stops if a positive IF solution of the reduced QBG is obtained, a DS solution is found, or just a single element remains. Thus no more than n such steps are needed.

Both IF and DS are polynomial time algorithms, therefore QSE is polynomial time algorithm, as well. The QSE heuristic is based on the assumption that performing the sequence of strategy eliminations we will not miss the equilibrium. Based on all (several dozen) random matrices we have tried for the IG models, the QSE has obtained the exact solution. Here are seven steps of QSE:

1) obtain a solution (x^*, y^*) of the LP problem (26)–(25): if a strictly positive solution of the system is found, go to Step 6;
2) search for an equilibrium in pure strategies using the Direct Search algorithm (32): if a solution is found, go to Step 6;
3) reduce the dimesion of LP problem by setting to zero the strategies $x_k = y_k = 0$ if the IF solutions $y_k \leqslant 0$ or $x_k \leqslant 0$ and no DS solution is found;
4) if a positive IF solution or a DS solution of the reduced system is found, go to Step 6;
5) if k is the only element in the reduced system set $x_k = y_k = 1$ and go to Step 6, otherwise go to Step 3;

6) test conditions (24) in terms of the initial problem with strategies that had been eliminated in Step 3 set to zero:
   if no – print 'No solution found',
   if yes – go to Step 7;
7) record the solution x_i^*, y_j^* as an equilibrium strategy.

3. Inspector Game (IG)

3.1. *Profit Functions*

Denote by $x = (x_1, \ldots, x_m)$, $x_i \geqslant 0$, $\sum_i x_i = 1$ the inspection vector and by $y = (y_1, \ldots, y_m)$, $y_j \geqslant 0$, $\sum_j y_j = 1$ the violation vector. Here x_i denotes the inspection probability of the object i and y_j means the violation probability of the object j. Denote by $u(i, j)$ the inspector's payoff when the object i is inspected and the object j is violated. Denote by $v(i, j)$ the violator's payoff when the object i is inspected and the object j is violated. Functions $U(x, y)$ and $V(x, y)$ denote the inspector's and violator's profit functions using inspection and violation vectors x, y. These vectors define probabilities of inspection and violation. Payoffs of IG

$$u(i,j) = \begin{cases} p_i g_i q_i, & \text{if } i = j, \\ 0, & \text{otherwise,} \end{cases} \tag{33}$$

and

$$v(i,j) = \begin{cases} -q_j p_i g_j + (1 - p_i) q_j g_j, & \text{if } i = j, \\ q_j g_j, & \text{otherwise.} \end{cases} \tag{34}$$

Here p_i is the probability of detecting the violation if it happens in the object i, q_i is the probability of the violation in the object i, and g_i is the payoff (potential) of the violation in the object i.

Expression (33) means that if the violation is completed and detected (for example, the prey is killed and a poacher is caught) then the inspector's premium is equal to the payoff of violation (the value of the killed prey). Expression (34) shows that if the violation is completed and detected, the violator's payoff is negative. The payoff is positive, if the violation is not detected.

The profit functions at given inspection and violation vectors x and y

$$U(x,y) = \sum_{i,j} x_i u(i,j) y_j, \tag{35}$$

$$V(x,y) = \sum_{i,j} x_i v(i,j) y_j. \tag{36}$$

Here $u(x, y) \neq v(x, y)$, thus the inspection model is a bimatrix game (Forgo *et al.*, 1999).

3.2. *Explicit Solution*

There exists explicit IF solution for a subset of IG. These solutions are useful for software testing.

Suppose, for example, that $q_i = g_i = 1$. Then from (33)–(34)

$$u(i,j) = \begin{cases} p_j, & \text{if } i = j, \\ 0, & \text{otherwise,} \end{cases} \tag{37}$$

and

$$v(i,j) = \begin{cases} -p_i + (1-p_i), & \text{if } i = j, \\ 1, & \text{otherwise.} \end{cases} \tag{38}$$

From here and expressions(13), (14), (6)

$$p_j y_j = U, \quad j = 1, \dots, m, \tag{39}$$

$$\sum_{i \neq j} x_i + (1 - 2p_j) x_j = V, \quad j = 1, \dots, m, \tag{40}$$

$$\sum_j y_j = 1, \quad \sum_i x_i = 1, \;\; y_j \geqslant 0, \;\; x_i \geqslant 0.$$

The solution is simple

$$y_i = x_i = 1 \Big/ \Big(p_i \sum_{k=1}^{n} 1/p_k \Big), \quad i = 1, \dots, n. \tag{41}$$

Note, that

$$x_i > 0, \quad y_j > 0, \;\; i, j = 1, \dots, m. \tag{42}$$

We see that if $g_i = q_i = 1$ then the IG problem (33)(34) is solved by the Irrelevant Fraud (IF) algorithm. The same is true if $g_i = g$, $q_i = q$, $i = 1, \dots, n$.

If no positive solution exist then we search for the equilibrium in pure strategies by the Direct Search algorithm (32). That complements the IF algorithm (29)–(25) and defines the equilibrium in pure strategies, if it exists. The next step is to apply the QSE algorithm.

3.3. *Preventing Corruption*

The game will not be played by rules if the players may win more by breaking them. Unauthorized deals (for example, bribes) are common tools for breaking the rules of non-zero-sum games where $v(i,j) \neq -u(i,j)$. An example of such deal is sharing the violator profit between the inspector and violator. Values of the optimal deal $(\bar{u}, \bar{v})$ are defined by the Nash bargaining conditions (Oven, 1995).

The optimal deal $(\bar{u}, \bar{v})$ depends on the set of feasible deals D, and on guaranteed profits defining what the players will get if the deal fails

$$u^* = \max_x \min_y U(x, y),$$
$$v^* = \max_x \min_y V(x, y). \tag{43}$$

Here

u^* is the maximal expected guarantee profit of the first player,

v^* is that of the second player.

In the Nash deal the first player gets $\bar{u}$ and the second obtains $\bar{v}$.

If exists a pair $(u, v) \in D$, such that $u > u^*$, $v > v^*$, then the optimal deal

$$(\bar{u}, \bar{v}) = \arg \max_{u,v} (u - u^*)(v - v^*), \tag{44}$$

where

$$(u, v) \in D, \quad u \geqslant u^*, \ \ v \geqslant v^*. \tag{45}$$

If the sum of expected profits of both players is restricted by c then

$$D = \big\{(u, v) \colon u + v \leqslant c\big\}, \tag{46}$$

and the Nash deal

$$(\bar{u}, \bar{v}) = \big((c + u^* - v^*)/2, \ (c + v^* - u^*)/2\big). \tag{47}$$

An obvious way to prevent unauthorized deals is by introducing an additional parameter w defining the expected penalty that makes the deal unprofitable.

The deal profits of players

$$u_d = \bar{u} - u^* = 1/2(c + u^* - v^*) - u^*,$$
$$v_d = \bar{v} - v^* = 1/2(c + v^* - u^*) - v^*. \tag{48}$$

We see that the profits are equal. Thus the minimal expected penalty

$$w^* = 1/2 \ (c - u^* - v^*). \tag{49}$$

Deal fails if the expected penalty $w > w^*$. Here $w = \omega \ p_\omega$ where ω is the deal penalty and p_ω is the probability of deal detection.

Assuming, that the deal of some IG is arranged before the game

$$c = \max_i \ g_i q_i. \tag{50}$$

Here c is the maximal expected payoff to be divided between the bargaining players.

3.3.1. *Example, Corrupt Inspector*
Suppose that $g_i = q_i = 1$ and the sum of expected profits is restricted by $c = \max_i g_i q_i = 1$.

From (41)

$$x_1^* = y_1^* = p_2/(p_1 + p_2),$$
$$x_2^* = y_2^* = p_1/(p_1 + p_2).$$

Then from (1), (2), (37), and (38)

$$u^* = p_1 p_2/(p_1 + p_2),$$
$$v^* = 1 - 2p_1 p_2/(p_1 + p_2) = 1 - 2u^*,$$
$$w^* = 1/2(1 - u^* - v^*) = 1/2(1 - u^* - 1 + 2u*) = 0.5u^*. \quad (51)$$

If, for example, $p_1 = p_2 = 0.5$ then from (49) the minimal expected penalty $w^* = 0.125$.

4. Experimental Software

4.1. *Short Description*

The software implementation of QSE , QSE-Java in short, is on the web (unsigned and signed versions of "Bimatrix Game" in the web-site section "Discrete Optimization") and is used for research cooperation and graduate studies. The QSE-Java is implemented as Java applets therefore QBG examples can be run by any browser supporting Java. In the unsigned version we enter data directly on the screen. That is not difficult for simple demonstration examples. In the signed version we can do that by files. Input of data by files is needed for large scale experimentation.

We start software description by unsigned version. Here the applet starts by clicking the line 'remote start'. This mode is not secure thus the input is made writing on the screen.

The payoffs $u(i, j)$ and $v(i, j)$ defined by (33), (34) in QSE-J are set in parametric form by entering input parameters. The meaning of the parameters P, G, Q in IG was explained in the comments of expressions (33), (34). Data can be changed by adding or deleting rows and entering the parameters P, Q, G.

Fig. 1 shows the input example for just two objects.

Parameters denoted as $Fine$ and $FineProbability$ define the penalty ω for unauthorized deals and the probability of their detection p_ω. To prevent unauthorized deals we enter the preferred penalties and their probabilities. The zero values mean no preferences thus the penalty is set automatically by expression (49). Note that apparently this is the first implementation of the Nash bargaining conditions (Oven, 1995) in bimatrix game software.

In QSE-Java the payoffs as functions of parameters are defined by clicking $(U, V)\ Specification$. The parametric form is converted into payoff matrices $u(i, j)$ and

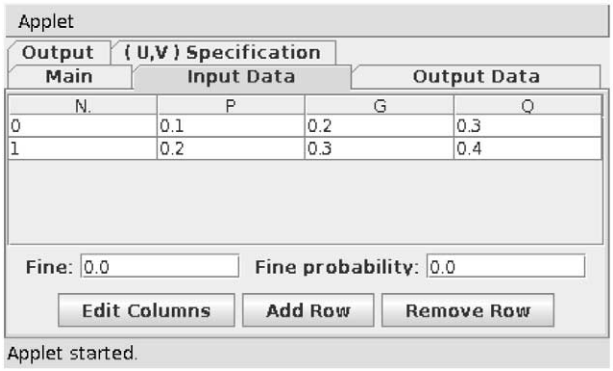

Fig. 1. Input of parameters P, G, Q.

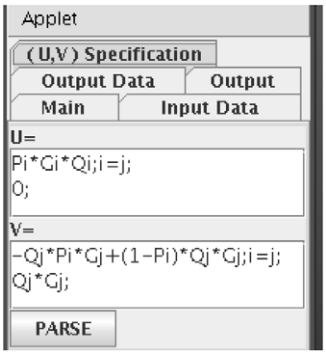

Fig. 2. Payoffs as functions of parameters in IG.

$v(i,j)$ by a parser during run-time. An example of these matrices is shown in the upper part of Fig. 4. Fig. 2 shows how the IG payoffs $u(i,j)$ and $v(i,j)$ depends on parameters Pi, Gi, Qi.

The optimization starts by selecting the $Main$ and clicking $Calculate$. The results can be observed in short and extended forms.

The short form is shown by clicking $Output\ Data$. It includes strategies $X0 = x_i, Y0 = y_j$ of NE. Fig. 3 shows the $Output\ Data$ when $n = 2$.

The extended form is opened by clicking $Output$. The $Output$ of the simplest example is in Fig. 4.

The upper part shows the payoff matrices $U = u(i,j)$ and $V = v(i,j)$ defined by expressions (33),(34) and calculated by the parser of QSE-Java. The lower part of Fig. 4 shows how the Direct Search 'DSA' works. First the indexes of maximal elements are defined by columns and by rows. Then the intersection (32) is calculated and the NE in pure strategies defined as the element $(1,1)$. This means that NE is obtained when both the inspector and the violator visit the same second object (in the QSE-Java object numbering starts from zero).

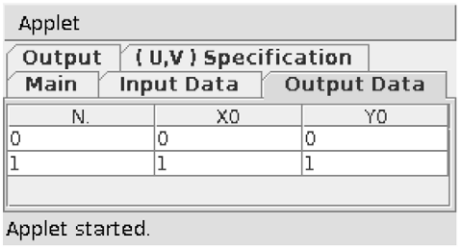

Fig. 3. Equilibrium strategies, $n = 2$.

The Direct Testing (24) is applied to QSE results by comparing the contract values U^* and V^* with maximal feasible profits $U\max$ and $V\max$. This time the equilibrium was found. The $Output$ ends by defining the minimal expected penalty (49) preventing unauthorized deal.

Using the unsigned version for input/output of data by large files the archive file 'remig05-19.tgz' should be downloaded and opened first. Then the applet is started by the command line that includes the policy file:

'appletviewer -J-Djava.security.policy=java.policy applet.html' (here 'applet.html' defines a html script to start QSE-Java).

The data can be entered by downloading an environmet file, for IG that is 'environment.txt':

```
V_EQUTION=-Qj*Pi*Gj+(1-Pi)*Qj*Gj;i\=j;\nQj*Gj;
U_EQUTION=Pi*Gi*Qi;i\=j;\n0;
DATA_SAVE_URL=file\:save_data.txt
DATA_LOAD_URL=http\://soften.ktu.lt/~mockus/inspsign/src/data.txt
VARIABLES=P G Q
RESULTS_SAVE_URL=file\:save_results.txt
```

This example shows how payoffs of IG (33) and (34) are defined in QSE-Java. However the software is designed for solving larger family of QBG defining them by different payoffs $u(i,j)$, $v(i,j)$ in the parametric form. The three variables P_i, G_i, Q_i are just for IG. For other games larger number of parameters can be employed. Note that no recompiling is needed while exploring different QBG.

The advantage of signed version is that data files can be entered by a browser if a user trusts the originating web-site. Therefore the input files should be ready for use. It would be convenient first to download the 'environment.txt' file by a browser, to be uploaded later, when the applet starts. Otherwise, a user prepares 'environment.txt' file in advance or writes '(U,V) Specification' on-line.

Note, that the web-site is updated at the end of each semester, as usual.

4.2. *Comparing with General Software Tools for Game Theory*

The Gambit (McKelvey *et al.*, 2005) is a popular software tool. That is a general software designed to solve different games. The advantage of Gambit is the implementation

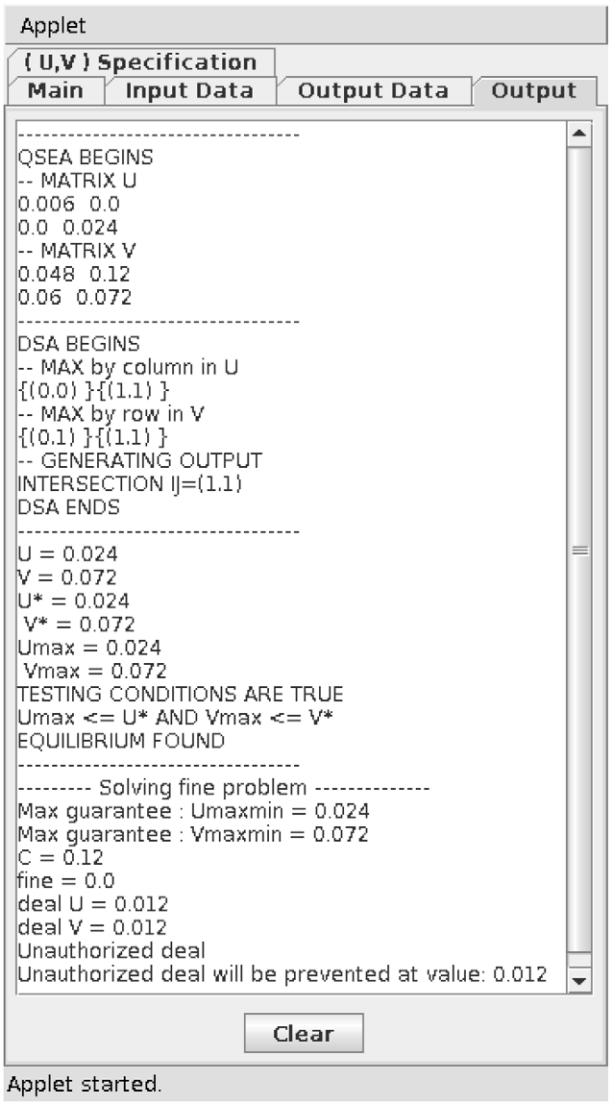

Fig. 4. Output, $n = 2$.

of exact algorithms and the convenient graphical user interface. Gambit displays a strategic game in the table form. Payoffs for each player are specified individually for each contingency, or collection of strategies, in the game. However analyzing large games by Gambit may become infeasible surprisingly quickly. Typically, the amount of time required to compute equilibrium increases rapidly in the size of the game.

Fig. 5 shows (by colored numbers) input and output of Gambit. Results of Gambit are obtained by the exact Lemke algorithm (Forgo *et al.*, 1999). The inner gray table shows output data of QSE algorithm. In this and other IG examples both the exact Gambit and

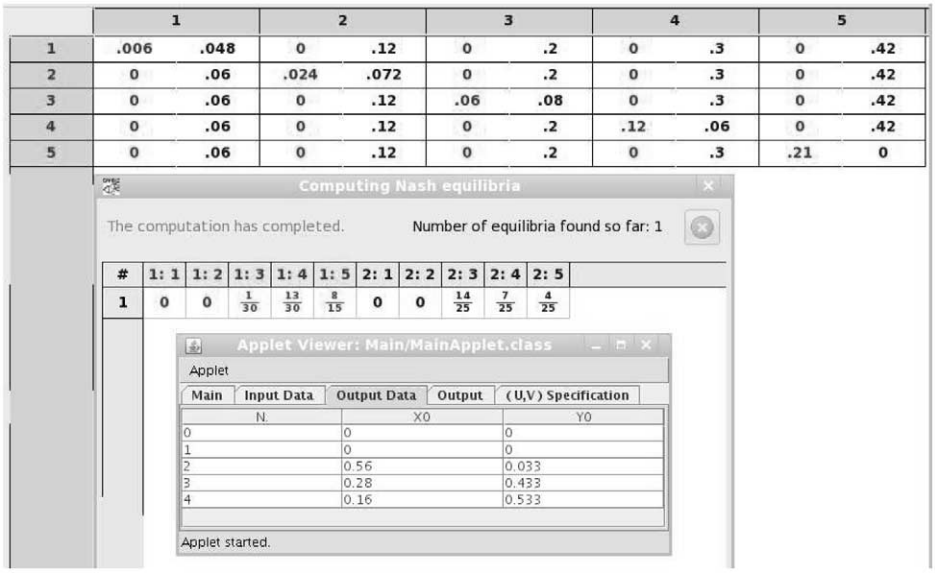

Fig. 5. Graphical user interfaces of Gambit and QSE.

the heuristic QSE-Java did find NE.

The aim of the QSE-Java is the web-based implementation of algorithms of polynomial complexity for the IG subset of QBG. This is done by Java applets. A feature of Java as compared with other platform-independent languages is the 'just-in-time' compiler. That, and some new improvements in Java 1.6 and Java 1.7 versions, makes Java web-based optimization algorithms almost as fast as the stand-alone versions implemented in the efficient languages such as C++.

The QSE-Java realizes a parser for transforming payoffs as some function's depending on several parameters into payoff matrices (33) and (34). In addition QSE-Java implements algorithms for prevention of unauthorized deals (49).

The main advantage of the QSE-Java is the simplicity of large scale experimentation. For the large scale experiments the random IG parameters P_i, G_i, Q_i, $i = 1, \dots, n$ were generated by uniform distribution in $[0, 1]$. The CPU time (by Intel(R), Core(TM)2, 2.4GHz, RAM 2GB, Linux, Fedora 8) was 20 sec. for the dimension $n = 200$, 1 min. for $n = 300$, 6 min. for $n = 400$, and 25 min. for $n = 800$. The NE was reached in all these experiments. Thus QSE guarantee equilibrium of QBG if it exists in pure or strictly mixed strategies (with no pure strategies) and provides NE for randomly generated IG examples, as usual.

Fig. 3 shows the $Output$ for $n = 2$.

4.3. *Application for Distance Graduate Studies*

The QBG model is simple. However, the model is based on fundamental results of games theory and operations research. That makes the model useful for studies of these topics.

The software is designed as an open-ended tool of research. Important task is to define a set of bimatrix games that can be solved by the polynomial algorithms. That includes defining proportion of randomly generated quadratic bimatrix games with Nash equilibrium in strictly mixed strategies. For equilibrium in pure strategies similar problem is solved by asymptotic formula (31). Solving the problem of equilibrium in strictly mixed strategies extensive experimentation could be useful. Numerical experiments applying QSE to different QBG can help evaluating advantages and disadvantages of QSE algorithm. Using and developing the software students better understand applications of Nash equilibrium (Nash, 1950).

These properties are useful for graduate studies where research skills are important. Students can easily create new bimatrix game models using the "Parser" mode implemented in the software, see Fig. 2. This way new features can be included and new situations investigated.

Implementation of the model as Java applet provides a cross-platform compatibility and makes the distances almost irrelevant.

The model is a part of a Web-based system of distance graduate studies. The main web-site:

`http://soften.ktu.lt/~mockus`,

and three mirrors, for reliability:

`http://pilis.if.ktu.lt/~jmockus`

`http://optimum2.mii.lt`

`http://kopustas.elen.ktu.lt/~jmockus`

includes this and accompanying optimization models.

Now the system is used regularly for distance graduate studies in two Lithuanian universities: Kaunas Technological University and Vilnius Technical University. The system was used during international graduate studies including Lappeenranta University of Technology, SF-53851, Lappeenranta, Finland (Heilo and Mockus, 2008).

5. Summary

We need polynomial time algorithms to solve large scale game problems and to explore numerically examples in studies of game theory. No polynomial time algorithm obtaining Nash Equilibrium (NE) is known for Bimatrix Games (BG) in general. Therefore, an important task is to define a subset of BG problems where NE can be reached in polynomial time.

This is done for Quadratic Bimatrix Games (QBG) by the Irrelevant Fraud (IF) model if NE exists in strictly mixed strategies (no pure strategies) or by the Direct Search (DS) if NE exists in pure strategies. Otherwise the heuristic polynomial time Quadratic Strategy Elimination (QSE) algorithm can be used. QSE includes both IF and DS, and the heuristic component. So far, we could not construct an Inspector Game (IG) example where the QSE does not find a solution, and in all our simulations with randomly generated IG matrices up to $n = 800$ the algorithm found NE.

A simple proof that the IF model defines the sufficient and necessary conditions for QBG if NE exists with no pure strategies is presented. The QSE algorithm is implemented as a Java applet and numeric simulations are conducted for IG from 200 to 800 objects. The CPU time ranged from 25 minutes for the largest dimensions, to 20 seconds for $n = 200$. Automatic testing for optimality included in QSE ensures that only correct solutions are accepted.

The algorithms are implemented as Java applets in the web-based system for scientific cooperation and studies. The source code is included as well. This helps teaching the game-theory aspects of the problem for the students with no programming skills and, more importantly, provides basic programming components in the game domain for the rest.

References

Berg, J. (2000). Statistical mechanics of random two-player games. *Physical Review E*, **61**, 2327–2339.

Conitzer, V., and T. Sandholm (2005a). Complexity of (iterated) dominance. In *Proceedings of the 6th ACM Conference on Electronic Commerce* (*EC-05*). Vancouver, Canada. pp. 88–97.

Conitzer, V., and T. Sandholm (2005b). A generalized strategy eliminability criterion and computational methods for applying it. In *Proceedings of the 20th National Conference on Artificial Intelligence* (*AAAI-05*). Pittsburgh, Pennsylvania, USA. pp. 483–488.

Dantzig, G. (1963). *Linear Programming and Extensions*. Princeton University Press, Princeton, N.J.

Forgo, F., *et al.* (1999). *Introduction to the Theory of Games*. Kluwer Academic Publishers.

Games (2006). *Game Theorists who have Received the Nobel Prize*. `http://lcm.csa.iisc.ernet.in/gametheory/nobel.html`.

Heilo, M., and J. Mockus (2008). Web based system for graduate studies: Optimization, games and markets. In *Progress in Industrial Mathematics at ECMI 2006*. Springer, Berlin.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, **4**, 373–395.

Knuth, D.E., *et al.* (1988). A note on strategy elimination in bimatrix games. *Operations Research Letters*, **7**, 103–107.

McKelvey, R.D., and A. McLennan (1996). Computation of equlibria in finite games. In *Handbook of Computational Economics*.

McKelvey, R.D., *et al.* (2005). *Gambit*: *Software Tools for Game Theory*. Version 0.2005.06.13. `http://econweb.tamu.edu/gambit/`.

McLennan, A., and J. Berg (2005). The asymptotic expected number of Nash equilibria of two player normal form games. *Games and Economic Behavior*, **51**, 264–295.

Mockus, J. (1997). A set of examples of global and discrete optimization: application of Bayesian heuristic approach I. *Informatica*, **8**(2), 237–264.

Mockus, J. (2003a). Development and investigation of a system for distance graduate studies and research cooperation in the internet evironment; applications in Lithuanian universities. In *Modelling and Simulation of Business Systems*, *BaltORS*. May 13–14, Vilnius, Lithuania. pp. 191–2003.

Mockus, J. (2003b). Stock exchange game model as an example for graduate level distance studies. *Computer Applications in Engineering Education*, **10**, 229–237.

Mockus, J. (2004). Walras competition model, an example of global optimization. *Informatica*, **15**, 525–550.

Mockus, J. (2006a). Investigation of examples of E-education environment for scientific collaboration and distance graduate studies, Part 1. *Informatica*, **17**, 259–278.

Mockus, J. (2006b). A system for distance studies and applications of metaheuristics. *Journal of Global Optimization*, **35**, 637–651.

Mockus, J. (2007). A system for distance graduate studies and research cooperation in the internet evironment and applications in Lithuanian universities. `http://pilis.if.ktu.lt/~jmockus`.

Murty, K. (2006). A new practically efcient interior point method for LP. *Algorithmic Operations Research*, **1**, 3–19.

Nash, J. (1950). Equilibrium points in n-person games. In *Proc. Nat. Acad. Sci. USA*, vol. 36. pp. 48–49.

Oven, G. (1995). *Game Theory*. Academic Press, San Diego.

Porter, R.W., *et al.* (to appear). Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior*.

Savani, R., and B. von Stengel (2004). Exponentially many steps for finding a Nash equilibrium in a bimatrix game. In *Foundations of Computer Science, Proceedings*. *45th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS'04*). October 17–19, Rome, Italy. pp. 258–267.

Smale, S. (1983). On the average number of steps of the simplex method of linear programming. *Mathematical Programming*, **27**, 241–267.

Von Stengel, B. (1998). Computing equilibria for two-person games. In *Handbook of Game Theory*, vol. 3. North Holland, Amsterdam.

J. Mockus graduated Kaunas University of Technology, Lithuania, in 1952. He got his doctor habilitus degree in the Institute of Computers and Automation, Latvia, in 1967. He is a principal researcher of the Systems Analysis Department, Institute of Mathematics and Informatics, Vilnius, and professor of Kaunas University of Technology. His research interests include global and discrete optimization.

Kvadratinio bimatricinio lošimo polinominio sudėtingumo modelis

Jonas MOCKUS

Tikslas – aprašyti, pagrįsti ir realizuoti interneto aplinkoje polinominio sudėtingumo algoritmą kvadratinių bimatricinių lošimų (QBG) Nash'o pusiausvyrai (NE) nustatyti. Siūlomas algoritmas jungia tiksliąją ir euristinę dalis.

Tikslioji dalis turi nejautrią apgaudinėjimui komponentę (IF), skirtą uždaviniams kai pusiausvyra egzistuoja be grynųjų strategijų. Kita, tiesioginio ieškojimo komponentė (DS), randa pusiausvyrą kai ji egzistuoja grynųjų strategijų aibėje. Heuristinė, kvadratinio heuristikų eliminavimo (QHE), dalis naudoja abi šias komponentes nuosekliai eliminuojant strategijas, neturinčias teigiamų sprendinių. Baigiant skaičiavimus nustatomos baudos už neteisėtus sandėrius.

Naujumas – tai būtinų ir pakankamų salygų įrodymas kai kvadratiniai bimatriciniai lošimai sprendžiami IF algoritmo pagalba bei QSE realizavimas interneto aplinkoje ir didelės apimties eksprimentų atlikimas uždaviniuose iki $n = 800$. QSE rado pusiausvyrą visuose skaičiavimuose. Tinklapis `http://pilis.if.ktu.lt/~jmockus` jungia šį ir kitus artimus optimizavimo modelius.