Comparison of the Hybrid and Lemke-Howson Methods for Solving Bimatrix Games

Evgeny G. Golshteyn Ustav Kh. Malkov Nikolay A. Sokolov

Central Economics and Mathematics Institute RAS
Nachimovky prospect 47,
117418 Moscow, Russia.
golshtn@cemi.rssi.ru, ustav-malkov@yandex.ru, sokolov@cemi.rssi.ru

Abstract

There exists a well-known efficient method for solving bimatrix games, namely, the Lemke–Howson method (LH). The proposed Hybrid method using a 2LP algorithm first obtains a point with a small Nash criterion value. Then it transforms this point to a starting point for the LH method. The numerical results of solving bimatrix games of the dimension up to 5000 are also presented.

The paper addresses the classical problem of the game theory: bimatrix games [Osborne, 2004]. After necessary preliminaries the 2LP method for solving bimatrix games, well known Lemke–Howson (LH) method [Lemke & Howson, 1964], its derivative algorithms (LH1 and LH2), and Hybrid method are described. The Hybrid method (with its steps 2LP, LH1 and LH2) was proposed as an alternative to the method of Lemke–Howson for finding Nash equilibrium. The second half of the paper is devoted to numerical comparison of these methods. At the conference OPTIMA-2013 [Golshtein et al., 2013] we presented the results of studies of an effective numerical method for finding approximate solutions of bimatrix games (2LP).

1 Bimatrix Games

Consider a bimatrix game Γ, where the players 1 and 2 operate with m and n strategies, respectively. Let $A = (a_{ij})$ and $B = (b_{ij})$ be the $(m \times n)$-matrices that determine the payoffs obtained by player 1 and 2 when the former uses strategy i $(1 \leqslant i \leqslant m)$ and the latter j $(1 \leqslant j \leqslant n)$, respectively.

The bimatrix game Γ in mixed strategies is defined with the aid of the above-mentioned matrices A and B as well as the strategy sets

$$X = \{x = (x_1, \dots, x_m) \in \mathbf{E}^m_+, \quad x^{\mathrm{T}} e_m = 1\}, \quad Y = \{y = (y_1, \dots, y_n) \in \mathbf{E}^n_+, \quad y^{\mathrm{T}} e_n = 1\}, \tag{1}$$

and the players' payoff functions

$$f_1(x, y) = x^{\mathrm{T}} A y, \quad f_2(x, y) = x^{\mathrm{T}} B y, \tag{2}$$

defined on the compact subset $Z = X \times Y$ of the Euclidean space $\mathbf{E} = \mathbf{E}^m \times \mathbf{E}^n$. Here and onward, for any $l > 0$, the subset $\mathbf{E}^l_+$ is the nonnegative orthant of the space $\mathbf{E}^l$, the vector e_l has all its components equal to 1. Since the payoff functions $f_i(x, y)$ $(i = 1, 2)$ are bilinear, the game Γ is convex, hence the set of its Nash points Z^* is nonempty (but not necessarily convex). The definition of convex games and their properties can be found in [Osborne, 2004].

In: Yu. G. Evtushenko, M. Yu. Khachay, O. V. Khamisov, Yu. A. Kochetov, V.U. Malkova, M.A. Posypkin (eds.): Proceedings of the OPTIMA-2017 Conference, Petrovac, Montenegro, 02-Oct-2017, published at http://ceur-ws.org

2 Nash Function

For the game Γ, one can introduce the standard (unnormed) Nash function as follows

$$F(z) = \max_{x \in X} f_1(x, y) - f_1(z) + \max_{y \in Y} f_2(x, y) - f_2(z), \qquad z \in Z. \tag{3}$$

Firstly such a function was introduced by H. Mills [Mills, 1960]. The inequality $F(z) \geqslant 0$ is true for all $z = (x, y) \in Z$. Moreover, $F(z) = 0$ if, and only if z is a Nash equilibrium or point of the game Γ. Therefore, the set of global minimum points of the function $F = F(z)$ on Z coincides with the set Z^* of the Nash points of the game Γ, with the global minimum value 0. In other words, solving the game Γ is tantamount to finding the global minimum of the function $F = F(z)$ over the subset Z, and the value of $F(z)$ can be treated as a natural measure of deviation of the point z from the set Z^*.

3 2LP-Method for Solving Bimatrix Games

The idea of consecutive solution of the LPs (see (4),(5) below) followed from the initial bilinear problem was firstly introduced by H. Konno [Konno, 1976]. For bimatrix games this idea was used by B. M. Mukhamediev [Mukhamediev, 1978] , A. V. Orlov and A. S. Strekalovsky [Orlov & Strekalovsky, 2005] and E. G. Golshtein et al. [Golshtein et al., 2013]. Even though the problem of minimization of the function $F(z)$ with respect to $z \in Z$ is extremely difficult because this function has a lot of local minima distinct from the global one, it is easy to reduce its (partial) minimization with respect to, e.g., the variable x (as part of the global variable $z = (x, y)$) while the other part (the variable y) is fixed, to a linear program. (The same is true if one decides to minimize F with respect to y with the first strategy x fixed.)

Indeed, let us fix the vector y (strategy of player 2) with values $y = y' = (y'_1, \ldots, y'_n) \in Y$ and consider the following linear programming problem $\mathcal{P}_1(y')$ with $m+n+1$ variables (x, v, α), where $v = (v_1, \ldots, v_n)$ is a vector of auxiliary variables, and with $n + 1$ equality constraints:

$$-x^{\mathrm{T}}(A + B)y' + \alpha \to \min, \quad x^{\mathrm{T}}B + v = \alpha, \quad x^{\mathrm{T}}e_m = 1, \quad x \in \mathbf{E}^m_+, \quad v \in \mathbf{E}^n_+, \quad \alpha \in \mathbf{E}^1. \tag{4}$$

Denote as $x^* = x^*(y')$ the first m components of an optimal solution of problem (4), then $F(x^*, y') \leqslant F(x', y')$ for every $x' \in X$.

Similarly, having fixed the first strategy x with some values $x = x' \in X$, one can introduce the linear program $\mathcal{P}_2(x')$ with $n + m + 1$ variables (y, u, β) (where $u = (u_1, \ldots, u_m)$ is a vector of auxiliary variables) and $m + 1$ equality constraints:

$$-(x')^{\mathrm{T}}(A + B)y + \beta \to \min, \quad Ay + v = \beta, \quad y^{\mathrm{T}}e_n = 1, \quad y \in \mathbf{E}^n_+, \quad u \in \mathbf{E}^m_+, \quad \beta \in \mathbf{E}^1. \tag{5}$$

Again, if $y^* = y^*(x')$ is the vector of the first n components of an optimal plan of problem $\mathcal{P}_2(x')$, then clearly $F(x', y^*) \leqslant F(x', y')$ for an arbitrary vector $y' \in Y$.

Now starting with an arbitrary *initial* strategy $y^0 \in Y$ of player 2, let us generate the sequence of iterations $\{z^t\} = \{(x^t, y^t)\}$, $t \geqslant 1$, as follows: x^{t+1} is a solution of $\mathcal{P}_1(y^t)$, while y^{t+1} solves problem $\mathcal{P}_2(x^{t+1})$, $t = 0, 1, 2, \ldots$

It is evident the inequality $F(z^{t+1}) \leqslant F(z^t)$ holds for all $t \geqslant 1$.

Let $\overline{T}$ be the maximum permitted number of iterations (pairs of tasks (4)-(5)), $\varepsilon_f > 0$ and $\varepsilon_F > 0$ be small numbers, S be the set of pure starting strategies. For any $z \in S$ define $t(z) = \min\{t \in \{1, \ldots, \overline{T}\}$: $F(z^t) - F(z^{t-1}) \leqslant \varepsilon_f\}$; We put $t(z) = \overline{T}$ at the failure of the inequality for all t. Denoting $Q(z) = F(z^{t(z)})$, we have $Q(S) = \min_{z \in S} Q(z)$ the lowest value that can be obtained. The game is approximately solved at the point z with accuracy ε_F if $F(z) \leqslant \varepsilon_F$. If $Q(S) > \varepsilon_F$ we believe that the solution to the game not found.

4 The Lemke-Howson Algorithm for Solving Bimatrix Games

The LH-algorithm replaces the solution of a bimatrix game by the search of a solution of the system of linear equations in a linear complementarity problem related to the game. By making the following change of variables $x := x/\alpha$, $y := y/\beta$ in the system (4)–(5), we obtain the system of linear equalities and inequalities (call it the LH-*problem*):

$$Ay + u = e_m, \quad x^{\mathrm{T}}B + v = e_n, \quad x, u \in \mathbf{E}^m_+, \quad y, v \in \mathbf{E}^n_+, \tag{6}$$

lacking the equalities $x^{\mathrm{T}}e_m = 1$, $y^{\mathrm{T}}e_n = 1$. Introduce the notation

$$H = \begin{pmatrix} O_m & A \\ B^{\mathrm{T}} & O_n \end{pmatrix}, \quad \begin{matrix} z = (z_1, \dots, z_{m+n}) \equiv (x, y), & e \equiv e_{m+n}, \quad E = \operatorname{diag}(e), \\ w = (w_1, \dots, w_{m+n}) \equiv (u, v), & o = (0, 0, \dots, 0) \in \mathbf{E}^{m+n}, \end{matrix},$$

where O_m and O_n are the square null matrices of dimension m and n, respectively; E is the unit matrix of dimension $m+n$, and thus yield the *linear complementarity problem*: Find a nonnegative ($z, w \in \mathbf{E}_+^{m+n}$) solution of the linear system

$$Hz + Ew = e, \tag{7}$$

such that $z_i w_i = 0$ for each $1 \leqslant i \leqslant m+n$ (complementarity conditions). Denote by $\Delta(z, w)$ the number of complementarity conditions broken at the point $(z, w) \in \mathbf{E}_+^{2(m+n)}$. We will say that a solution (z, w) of system (7) satisfies *almost all* complementarity conditions if $z_i w_i \neq 0$ for only one index i, $1 \leqslant i \leqslant m+n$.

The initial basis of the LH-problem is composed of all slack variables w. The latter satisfies the complementarity conditions, however, $z = o$, which means that the equalities $x^{\mathrm{T}}e_m = 1$, $y^{\mathrm{T}}e_n = 1$ don't hold. By having introduced into the basis one of the structural variables (say, z^1) and violated therewith one of the complementarity conditions, the LH-method starts from the initial point (z^1, w^1) and makes further steps (iterations) of the simplex-method-type until the broken complementarity condition becomes true. The generated sequence of vertices (satisfying almost all complementarity conditions) forms the so-called *Lemke path* running the feasible bases of system (7). After several iterations of the LH-method, the structural variables x and y will boast various non-zero entries. One can make the inverse transformation of the variables back to their original form and continue the Lemke steps by pivoting the bases of the initial problem taking into account the constraints $x^{\mathrm{T}}e_m = 1$, $y^{\mathrm{T}}e_n = 1$. In this manner, we exclude the possibility of returning to the initial point $z = o$.

The algorithm stops if either it has run into a previously generated point (that is, a loop has been closed) or the current point satisfies all complementarity conditions, i.e., it solves the linear system. As a rule (but not always) the algorithm finds an exact solution of system (7), hence, it solves the game Γ. In order to come back to the original variables, one must normalize the just obtained solution of the system as follows: $x := x/(x^{\mathrm{T}}e_m)$, $y; = y/(y^{\mathrm{T}}e_n)$.

The LH-method (as well as its derivative algorithms LH1 and LH2) as steps of Hybrid algorithm is founded on the LH-*procedure* that generates a sequence of pivoted bases under the **principal rule**: the leaving variable is selected so that the other basic variables keep having nonnegative values, and the entering variable is either z_k or w_k ($1 \leqslant k \leqslant n+m$), which is complementary to the leaving variable (that is, w_k or z_k, respectively).

Thus, the LH-procedure determines the Lemke path uniquely after the starting structural variable has been selected to introduce into the initial basis. The pitfalls of the LH-method are, on the one hand, a possibility of loops (returns to the initial point), and on the other hand, a forbiddingly large number of iterations. If no solution has been found for the selection of $z = z^1$ as the initial point, we start the method again with the initial $z = z^2$, and so on. The game is considered as unsolved if no solution is found after all the initial points z^i, $i = 1, \dots, n+m$, have been tried.

5 The LH1-Algorithm with a "Hot Start"

We say that a point $z = (x, y) \in \mathbf{E}^{m+n}$ *approximately* solves the bimatrix game Γ, whenever $F(z) \leqslant \varepsilon_F$ and $\Delta(z, w) \leqslant 1$ (recall that $\Delta(z, w) = 0$ at the exact solution of Γ). Next, we say that a point $z = (x, y)$ is *promising* if $\varepsilon_< F(z) \leqslant \overline{F}$ and $1 \leqslant \Delta(z, w) \leqslant \overline{\Delta}$, where ε_F, $\overline{F}$, and $\overline{\Delta}$ are some preliminary selected parameters.

Let z be a promising starting point for the game Γ. With the former, we start the LH-method's version that finds the solution (Nash point) with a lower number of broken complementarity conditions. This version is named as the LH1-*algorithm*.

In contrast to the "classical" LH-method where the unit matrix determines the initial basis and the complementarity condition $\Delta(z, w) = 0$ holds for $z = o$, $w = e$, in the LH1-algorithm, $\Delta(z, w) = k > 0$, i. e., the initial basis comprises (in an arbitrary order) $2k$ basic variables z_i, w_i, $i = i_1, \dots, i_k$ (called "offenders"), for which $z_i w_i > 0$. At the same time, there are $2k$ nonbasic variables ("candidates" for entering the basis) such that $z_j = w_j = 0$, $j = j_1, \dots, j_k$. In order to diminish the number of broken complementarity conditions, some of the candidate variables are to enter the basis to replace the offenders.

An initial basis of an LH-problem for a "hot start" is combined from the bases (without variables α and β) for problems (4) and (5), obtained by the 2LP-method, what we use as first step of Hybrid algorithm.

We try to introduce all the candidate variables one by one instead of the offenders. If we have managed to do that with success and all the complementarity conditions have been satisfied, then the LH-problem (6) has been solved. Otherwise, that is, when entering a candidate variable into the basis, the leaving basic variable doesn't break the corresponding complementarity condition (i.e., it isn't an offender), then we again use the LH-procedure and introduce the variable complementary to the variable that has just left the basis. We repeat these operations until either the offender has been expelled from the basis and Δ_0 has dropped, or the number of those attempts has reached the upper limit of $\overline{I}(m+n)$, where the constant $\overline{I}$ has been preliminary selected. It is also possible that a loop closes before the upper limit has been achieved: in this case, we simply switch to the next candidate variable without changing the value of Δ.

In our test numerical experiments, in the majority of the considered examples, the proposed algorithm did manage to exclude from the basis one variable from each pair of offenders, that is, all the broken complementarity conditions were successfully repaired. If there was left broken complementary conditions we go to step 6 as third step of Hybrid algorithm.

6 The LH2-Algorithm with a Supplementary Column

If the LH1-algorithm has failed with excluding all offenders from the basis, we will try to do that with the aid of a so-called *supplementary column*.

First, we will forcefully replace one of the variables in each offending pair by a candidate variable from the list of candidate pairs with the aim of repairing the $\overline{k}$ broken complementarity conditions that the LH1-method has failed to remove. Let $k_x \geqslant 0$ and $k_y \geqslant 0$ be the numbers of the invalid complementarity conditions related to the variables x and y, respectively.

If $k_x > 0$ then among all possible k_x^2 candidates for entering the basis we select the variable that maximizes the minimum negative value of the current basic variables among such negative basic values that must appear after the forceful change of the basis. Now we update k_x by subtracting 1 and repeat the procedure until we reach $k_x = 0$. The similar actions are undertaken towards $k_y > 0$.

Having selected the next candidate variable, decompose the corresponding column with respect to the current basis. Among the coefficients of this decomposition, find those belonging to the positions related to the offenders. Detect the maximum (or large enough) absolute value among the just selected coefficients and introduce the candidate variable exactly in this position.

A supplementary column should be generated in such a way that as a result of having introduced it into the basis, all the negative values of the basic variables disappear. In order to do that, we compose the supplementary column as a sum of the columns that correspond to the variables having negative values in the current basic solution. This new column multiplied by the minimum negative weight (that is, having the maximum absolute value among the above mentioned negative values) is introduced into the basis and occupies the position corresponding to the same basic variable with the most negative value. It is easy to check that after that, all basic variables will acquire positive values, while the variable related to the supplementary column will get the value 1.

Next, we begin the LH-procedure starting from the variable complementary to the variable that has left the basis when introducing into it the supplementary column. We repeat the steps of the LH-procedure until either the supplementary column leaves the basis (that is, a Nash point has been obtained), or the number of iterations exceeds the upper limit.

7 The Hybrid Method (Hyb-Algorithm) for Solving Bimatrix Games

Now that all the necessary parts (the 2LP-method, LH1- , and LH2-algorithms) have been described we are in a position to present our Hyb-Algorithm for solving bimatrix games as a solid alternative to the LH-method and even as a possible tool for finding a solution to the game when the LH-method fails.

$$\begin{tikzcd}
 & & & 1. & 1. & \\
0. \arrow[r, Rightarrow] & 1. \arrow[r, Rightarrow] \arrow[d, Rightarrow] & 2. \arrow[r, Rightarrow] & 3. \arrow[r, Rightarrow] \arrow[u, Rightarrow] \arrow[d, Rightarrow] & 4. \arrow[r, Rightarrow] \arrow[u, Rightarrow] & \textit{stop} \\
 & \textit{stop} & & \textit{stop} & &
\end{tikzcd}$$

Scheme of hybrid algorithm

Step 0. Define the start point, for example, the pure strategy $y_1 = 1, y_i = 0, i = \overline{2, n}$.

Step 1. Going through the set of initial points (pure strategies) look for a prospective starting point ($F(x, y) \leqslant \varepsilon_F$) by an 2LP algorithm, otherwise use a new start (pure strategy) point. If the set of starting points has exhausted, the game Γ is reported as unsolved.

Step 2. Collect the basis of LH problem by making use of the solution obtained by the 2LP-algorithm. Detect the broken complementarity conditions and list the offenders (the basic variables in broken complementary conditions) and candidates (the pairs of nonbasic variables related by the complementary conditions).

Step 3. (**Key**) Attempt to remove the broken complementary conditions by introducing one candidate from each pair as starting points into the basis in LH (LH1) method with **"hot start"**. If in the end, all the offenders have been excluded from the basis, a Nash point has been obtained and the game solved. Otherwise, go to Step 4 to generate a supplementary column.

Step 4. It will get a equilibrium point by using the supplementary column if step 3 fails. In all pairs offenders will forced replace one of them by one of the candidates. In the obtained solution will be negative values. An supplementary column is collected as the sum of columns that correspond to variables with negative values in the basic solution. This column is used as starting point for LH (LH2) method with **"hot start"**.

If the solution was not obtained through steps 1, 3 or 5, repeat from 1 step.

8 Numerical Experiments

The tested algorithms (2LP, LH, and Hyb) were coded and implemented with the software MATLAB 7.14.0.739 R2012a. The linear subprograms were solved by the procedure of `cplexlp` from the IBM ILOG CPLEX 12.6.2, which is much more efficient than the procedure `linprog` of MATLAB. The personal computer IntelCore i5-2400 CPU (3.1 GHz), 4 Gb RAM was used for the computation in the majority of cases.

The algorithms were tested and their efficiency was compared mainly on the base of a family of bimatrix games containing 15 series of medium-size games with the parameters $m, n \in \{20, 40, 60, 80, 100\}$, $m \leqslant n$. For each series, $\overline{R} = 100$ of problems were initially generated. The main parameters of the algorithm(s) (if not specified otherwise) are as follows: $S = m$, $\varepsilon_F = 10^{-4}$, $\overline{F} = 0.01$, $I = \overline{\Delta} = 5$, $\overline{I} = 1$.

With the purpose of saving the efforts and running time, the ordered set $X^0 = \{e_m^1, \dots, e_m^m\}$ of pure strategies of player 1 was selected as the set of possible starting points. Here, e_m^i is the standard unit vector of the space $\mathbf{E}^m$ having all entries zero except entry i, which is 1.

The matrices $A = (a_{ij})$ and $B = (b_{ij})$ were generated in two stages. First, the entries a'_{ij} and b'_{ij} were generated independently by the MATLAB generator of pseudo-random numbers uniformly distributed in the interval $(0, 1)$. After that, we applied the transformation $a_{ij} = a'_{ij} + \varphi b'_{ij}$, $b_{ij} = b'_{ij} + \varphi a'_{ij}$, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, where φ is the coefficient of the *matrix mutual interdependency*, $0 \leqslant \varphi \leqslant 1$. The latter transformation implies that the entries of both matrices lie within the range of $(0, 1 + \varphi)$.

The numerical experiments investigated the performance and compared the efficiency of the algorithms 2LP, LH, and Hyb when solving bimatrix games. We also examined whether it was possible, starting with the 2LP method, to obtain such a starting point for the LH-algorithm that would help find an exact solution of the game with a save on the computational efforts (iterations, running time, etc.).

The numerical and testing results are given in the three tables below, where, in Tables 1–2, for the reader's convenience, the results are shown for 5 series of games, while the lines Sum provides the numbers obtained by summing up over all the series of games.

The games having solutions in pure strategies needn't be solved by the LH and Hybrid methods; Since analyzing preliminary payoff matrixes we can simply establish whether the game has Nash equilibrium in pure strategies.

Table 1 accumulates the testing results obtained for the family of 1500 games treated by the LH-method for three values of the matrix mutual interdependency parameter ($\varphi \in \{0; 0.1; 0.2\}$). The notation of Table 1 means: m, n are the game's dimension parameters; the parameter I determines the upper limit (equal to $I\,(m + n)$) of iterations allowed in the LH-algorithm;k_{PS} is the number of games having solutions in pure strategies k_{LH} is the number of games solved by the LH-algorithm; k_{NO} is the number of games on which the LH-algorithm failed. To resume, $k_{\text{PS}} + k_{\text{LH}} + k_{\text{NO}} = 100$; In addition, for each unsolved game, the total number of tested starting points was m and the total number of iterations for every starting point didn't exceed $I(m + n)$; the value of S_{sum} equals the total number of starting points tested, J_{LH} is the total number of simplex-type iterations, Time is the running time (in seconds) that the LH-method spent to find the solution, $S_{\max}$ is the maximum number of the starting points tested when solving one (solved) game.

Analysis of the data from Table 1 allows one to make the following conclusions.

1. Generation of the independent matrices A and B gets 935 games solvable in the pure strategies. Therefore, when $\varphi = 0$ one needs to solve only 565 games. For $\varphi = 0.1$ we had to solve 1477 games, for $\varphi = 0.2$ we treated

1496 games. For $\varphi \leqslant 0.1$, ALL the games from the family were solved by the LH-algorithm, while for $\varphi = 0.2$, the LH-method failed on 25 games of the total of 1500.

2. The value of the parameter I determining the maximum allowed number of iterations when solving a game seriously affects the efficiency of the LH-method. When selecting an appropriate value for this parameter I, one should take into account the value of φ as well as the dimension of the solved problem. For our test examples, the following values proved to be well-suited (but not necessarily optimal): $I = 0.5$ for $\varphi = 0$ and $I = 5$ for φ either 0.1 or 0.2.

Our test computations have proven that it makes sense to classify the initial points z^i, $i = 1, \dots, m+n$, by trying to produce no more than $I(n+m)$ iterations with each one, where $0.01 < I \leqslant 5$. As one can see from the numerical results given below, the efficiency of the algorithm vitally depends on the constant value of I, which is desirable to select particularly for each game.

Table 1: Solution Results Obtained by the LH-algorithm Tested on 100 Games for Various Values of φ

φ	I	Num	Sizes	Result			Total amount			Time
			$m = n$	k_{PS}	k_{LH}	k_{NO}	S_{sum}	$S_{\max}$	J_{LH}	sec
0	0,5	1	20	63	37	0	68	6	1047	1
		2	40	61	39	0	87	8	2704	5
		3	60	67	33	0	115	23	5837	10
		4	80	65	35	0	99	10	6125	13
		5	100	56	44	0	142	10	11242	25
		Sum		935	565	0	1364	—	66723	129
0.1	5	1	20	12	88	0	88	1	3829	3
		2	40	0	100	0	140	8	28901	29
		3	60	0	100	0	218	9	96196	124
		4	80	0	100	0	345	18	232845	377
		5	100	0	100	0	525	22	473061	935
		Sum		23	1477	0	3210	—	1744759	2894
0.2	5	1	20	4	96	0	97	2	5521	4
		2	40	0	100	0	158	9	42513	171
		3	60	0	100	0	510	25	279997	349
		4	80	0	100	0	1358	61	1050087	1656
		5	100	0	84	16	4075	100	4039640	7782
		Sum		4	1473	25	11860	—	9487817	16705

Table 2 presents the test results for solving the same family of 1500 games with the aid of the Hyb-algorithm for the same three distinct value of the matrix interdependence parameter $\varphi \in \{0;\ 0.1;\ 0.2\}$. The following notation is accepted: m, n are the game's dimension parameters;where k_{PS} is the number of games solvable in the pure strategies, such games needn't the Hyb-method to solve them; k_{2LP}, k_{LH1}, k_{LH2} are the numbers of the games solved by the 2LP-, LH1-, and LH2-methods; k_{NO} the number of the games on which the Hyb-algorithm failed. Therefore, $k_{\text{PS}} + k_{\text{2LP}} + k_{\text{LH1}} + k_{\text{LH2}} + k_{\text{NO}} = 100$; We also denote by S_{sum} the total number of the used starting points, S_{Hyb} is the total number of the promising starting points used, J_{2LP} is the total number of iterations (i.e., the pairs of solved linear programs), performed by the 2LP-algorithm, $J_{\text{LH}} = J_{\text{LH1}} + J_{\text{LH2}}$ sums the total numbers of iterations performed by the algorithms LH1 and LH2, Time is the total running time (in seconds) that the Hyb-method spent to solve the games in question.

Based upon the data from Table 2, the following conclusions can be drawn.

1. Just as the LH-method, the Hyb-algorithm needed to be applied to only 565 games when $\varphi = 0$, to 1477 games for $\varphi = 0.1$, and to 1496 games when $\varphi = 0.2$. For $\varphi \leqslant 0.1$, the Hyb-algorithm solved all the games from the tested list, for $\varphi = 0.2$ it failed in 60 games from 1500, which is 2,6 times more than happened to the pure LH-method.

2. Each series of games admitted all three solution modes: for $\varphi = 0$ there occurred $k_{\text{2LP}} = 187$, $k_{\text{LH1}} = 313$,

Table 2: The Results Shown by the Hyb-Algorithms When Solving the Test Games for Various Values of φ

φ	Num	Sizes	Result					Total amount				Time
		$m = n$	k_{PS}	k_{2LP}	k_{LH1}	k_{LH2}	k_{NO}	S_{sum}	S_{Hyb}	J_{2LP}	J_{LH}	sec
0	1	20	63	18	16	3	0	69	53	222	4607	47
	2	40	61	12	22	5	0	86	73	310	10648	72
	3	60	67	9	18	6	0	86	80	293	12937	78
	4	80	65	14	17	4	0	82	80	302	15192	80
	5	100	56	11	25	8	0	118	114	431	22045	119
	Sum		935	187	313	65	0	1282	1154	4665	187650	1175
0.1	1	20	12	33	41	14	0	171	132	585	13973	120
	2	40	0	36	43	21	0	337	260	1355	54860	300
	3	60	0	25	51	24	0	508	406	2298	120700	529
	4	80	0	30	56	14	0	787	599	3752	238982	953
	5	100	0	25	49	26	0	939	644	4727	338795	1196
	Sum		23	429	760	288	0	7440	5687	33794	1910344	8923
0.2	1	20	4	21	49	26	0	264	177	1014	28583	204
	2	40	0	34	44	22	0	629	436	3040	149248	658
	3	60	0	26	47	26	1	1723	988	9288	624208	2115
	4	80	0	30	49	14	7	2327	1059	13842	1157629	3235
	5	100	0	33	42	12	13	3988	1451	25706	2563992	6166
	Sum		4	387	717	332	60	20960	10625	120343	9504944	28147

$k_{\text{LH2}} = 65$; for $\varphi = 0.1$ there happened $k_{\text{2LP}} = 429$, $k_{\text{LH1}} = 760$, $k_{\text{LH2}} = 288$; for $\varphi = 0.2$ there appeared $k_{\text{2LP}} = 387$, $k_{\text{LH1}} = 717$, $k_{\text{LH2}} = 332$.

3. The average number $\widetilde{S}$ of the used starting points for one solved problem for $\varphi = 0;\ 0.1;\ 0.2$ was, respectively, 2.3; 5; 14. As φ grows the ratio $S_{\text{Hyb}}/S_{\text{sum}}$ of the promising starting points to the total number of the (used) starting points drops as follows: $(0.9; 0.8; 0.5)$. The total running time needed to solve all tested games prove to be 20, 138, and 469, respectively.

Table 3 provides the numerical results obtained by the LH- and Hyb-algorithms when solving several bimatrix games of a large size. Here, the following notation is used: the parameter I determines the upper bound (equal to $I(m+n)$) for the number of iterations allowed to produce by the LH-algorithm; m, n define the size of the bimatrix game; ϱ is the number of the games in the serie, while k_{NO} denotes the number of unsolved games in the series in question. S_{sum} gives the total number of the used starting points, J_{sum} is the total number of the simplex-type iterations conducted, and Time denotes the aggregate running time (in seconds).

Having analyzed the data of Table 3, we come to the following conclusions.

1. Once again, the results have shown that a lucky choice of the parameter I greatly affects the velocity of the LH-algorithm. For instance, when solving the 400×800 game, for the value of $I = 500$ (that is, the upper bound of iterations equalling $I(m+n) = 600000$) the totality of iterations (substitutions of the basis) resulted in $4 \times 600000 + 221 = 2400221$, whereas for $I = 0,25$ only $4 \times 0,25(400 + 800) + 221$ (iterations on the 5-th start point) $u = 1421$ were made.

In both cases, the final result was achieved after the fifth starting point has been employed and altogether 221 iterations has been made, but the difference in the running time was enormous. However, the Hyb-algorithm made use of only three starting points, and the final solution was obtained by the LH1-procedure, after 1464 iterations performed for 17 seconds.

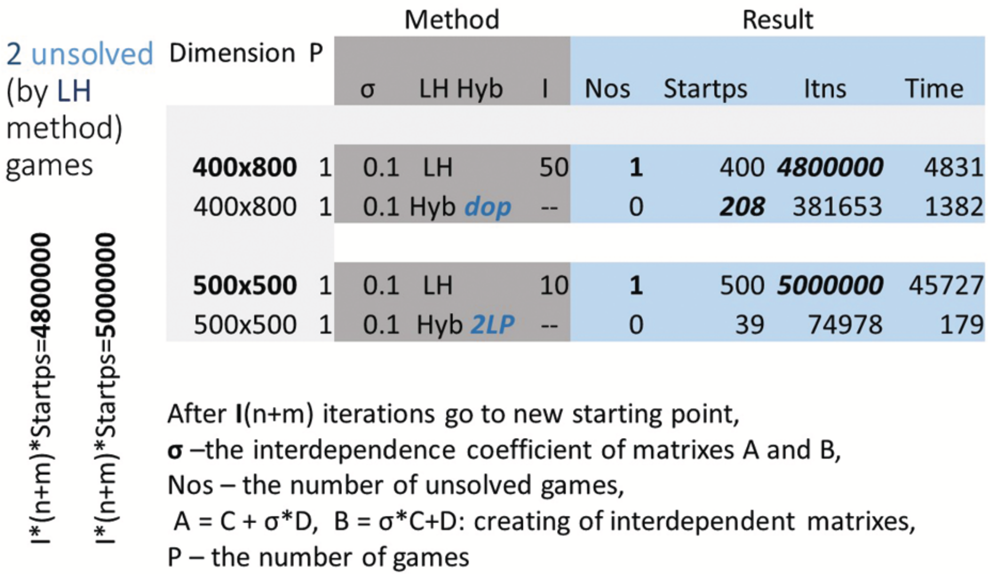
2. When applied to the two games listed in Table 3 with the dimensions 400×800 and 500×500, both with the value of $\varphi = 0.1$, the LH-algorithm failed for various values of the parameter I. At the same time, these games were solved successfully with the Hyb-method: the game of the size 400×800 was solved after having checked 208 starting points and conducted 381653 iterations for1382 seconds; the solution of other game of the

Table 3: Comparison of the Results Obtained by the LH- and Hyb-Algorithms Solving Big-Sized Bimatrix Games

Num	Sizes		ϱ	φ	Method		Result			Time
	m	n			LH or Hyb	I	k_{NO}	S_{sum}	J_{sum}	sec
1	400	800	1	0	LH	500	0	5	2400221	6051
	400	800	1	0	LH	0.25	0	5	1421	19
	400	800	1	0	Hyb	–	0	3	1468	17
2	400	800	1	0.1	LH	50	1	400	4800000	4831
	400	800	1	0.1	Hyb	–	0	208	381653	1382
3	500	500	1	0.1	LH	10	1	500	5000000	45727
	500	500	1	0.1	Hyb	–	0	39	74978	179
4	3000	3000	39	0	LH	0.01	0	6004	358826	19374
	3000	3000	39	0	Hyb	–	0	328	485359	17238
5	5000	5000	5	0	LH	0.01	0	880	87732	13566
	5000	5000	5	0	Hyb	–	0	66	118317	27499

dimensions 500×500 was similarly finished after having tried 39 starting points and applied 74978 iterations for 179 seconds.

3. Both algorithms efficiently solve the games with independent matrices at about the same computational cost provided that the value of the parameter I has been appropriately selected. For instance, when solving the problem of the size 3000×3000 the Hyb-algorithm made use of 328 starting points and performed 485359 iterations for 17238 seconds. The LH-method had to check 6004 starting points and perform 358826 iterations for 19374 seconds, with the value of the parameter $I = 0.01$ (the upper bound for the number of iterations thus being $I\,(m + n) = 60$).



2 unsolved (by LH method) games

I*(n+m)*Startps=4800000

I*(n+m)*Startps=5000000

		Method			Result			
Dimension	P	σ	LH Hyb	I	Nos	Startps	Itns	Time
400x800	1	0.1	LH	50	**1**	400	***4800000***	4831
400x800	1	0.1	Hyb *dop*	--	0	***208***	381653	1382
500x500	1	0.1	LH	10	**1**	500	***5000000***	45727
500x500	1	0.1	Hyb *2LP*	--	0	39	74978	179

After **I**(n+m) iterations go to new starting point,
σ –the interdependence coefficient of matrixes A and B,
Nos – the number of unsolved games,
A = C + σ*D, B = σ*C+D: creating of interdependent matrixes,
P – the number of games

Figure 1: Two problems, unsolved by LH method, but solved with a hybrid algorithm.

You can see on Figure 1. the results of solving two problems (with interdependent matrices) by LH method and the hybrid algorithm. LH method **failed** to solve these problems using all the initial points (400 and 500, respectively). The hybrid algorithm got the solution of the first problem on the 208-th start point using the additional column, and the second problem was solved on the 39th starting point on the local search step.

9 Conclusions

All the above presented leads to the following conclusion: The proposed Hyb-algorithm has proven to be competitive as compared to the 2LP-method and LH-algorithm. Moreover, in some instances, the Hyb-method manages to solve the games on which both the 2LP-method and LH-algorithm fail.

References

[Osborne, 2004] M. Osborne (2004). *An Introduction to Game Theory.* New York: OxfordUniversity Press.

[Mills, 1960] H. Mills (1960). Equilibrium points in finite games. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):397–402.

[Golshtein et al., 2013] E. G. Golshtein, U. Kh. Malkov, N. A. Sokolov (2013). A Numerical Method for Solving Bimatrix Games (2LP). *In Proceedings of OPTIMA'2013 (Petrovac, Montenegro September 22 - 28, 2013)*, p. 67

[Golshtein et al., 2013] E. G. Golshtein, U. Kh. Malkov, N. A. Sokolov (2013). A Numerical Method for Solving Bimatrix Games. *Economica i Matematicheskie Metody*, 49(4):94–104 (In Russian).

[Konno, 1976] H. Konno, (1976). A cutting plane algorithm for solving bilinear programs. *Mathematical Programming*, 11:14–27.

[Mukhamediev, 1978] B.M. Mukhamediev (1978). On solving bilinear programming problems and cutting off. All equilibrium situations in bimatrix games. *Zh. Vychisl. Mat. Mat. Fiz.*, 18:351–359 (in Russian).

[Orlov & Strekalovsky, 2005] A. V. Orlov, A. S. Strekalovsky, (2005). Numerical search for equilibria in bimatrix games. *Computational Mathematics and Mathematical Physics*, 45:947–960.

[Lemke & Howson, 1964] C. E. Lemke, J. T. Howson, Jr. (1964). Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12:778–780.