The Lemke–Howson Algorithm Solving Finite Non-Cooperative Three-Person Games in a Special Setting

Evgeny Golshteyn¹, Ustav Malkov¹, and Nikolay Sokolov¹

¹Central Economics and Mathematics Institute of Russian Academy of Science, Nakhimovsky Prospect 32 117418 Moscow, Russia

ustav-malkov@yandex.ru, sokolov_nick@rambler.ru

Keywords:  Bimatrix game, Hexamatrix game, Function, Mixed strategies, Nash equilibrium, Lemke–Howson algorithm

Abstract. We present a brief outline of an approximate method (3LP) proposed by E.G. Golshteyn for solving three-person games in mixed strategies. Similar to the 2LP-algorithm (which approximately solves bimatrix games), the solution procedure consists in the search of a global minimum of the so-called Nash function. By making an exhaustive search of the pairs of the initial strategies the algorithm 2LP (3LP) finds an exact solution of the game if the condition of reciprocal complementarity holds. The numerical experiments show that the 2LP-method successfully competes with the Lemke–Howson (LH) algorithm, which efficiently solves the bimatrix games. Unfortunately, the LH-algorithm cannot be applied to solve arbitrary three-person games. However, we have adapted the Lemke–Howson method to the solution of a special setting called the hexamatrix games. We have also conducted a thorough testing of the LH-algorithm to reveal its advantages and minor points as well.

Introduction

In [1] an algorithm approximately solving finite non-cooperative three-person games (3LP) was proposed. The testing results illustrating the efficiency of the said method’s application can be found in [2]. In this paper, we have tested the 3LP-algorithm as applied to a special setting of the 3-persons game [7]. Moreover, we have managed to adapt the classical Lemke–Howson (LH) method [4] to the solution of the 3-persons game in the above-mentioned special setting. Let us specify both the general and the special settings of the 3-persons games.

The Three-Person Game in the General Setting

A finite non-cooperative 3-persons game $\Gamma$ is defined with three sets $X$, $Y$, $Z$ of strategies of the first, second, and third player respectively, where $X = \{x = (x_1, \ldots, x_m)^T \in \mathbb{E}^m : x^T e_m = 1, x \_ o_m \}$, $Y = \{y = (y_1, \ldots, y_n)^T \in \mathbb{E}^n : y^T e_n = 1, y \_ o_n \}$, $Z = \{z = (z_1, \ldots, z_l)^T \in \mathbb{E}^l : z^T e_l = 1, z \_ o_l \}$.
\[ z \geq o_l \}, \text{ together with their payoff functions as follows} \]

\[ f_x(\omega) = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{l} a_{ijk} x_i y_j z_k, \]

\[ f_y(\omega) = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{l} b_{ijk} x_i y_j z_k, \]

\[ f_z(\omega) = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{l} c_{ijk} x_i y_j z_k. \]

Here, one has \((a_{ijk}), \ (b_{ijk}), \ (c_{ijk})\) — the players’ 3-dimensional payoff tables (without any loss of generality one can assume that all the entries of those tables are positive real numbers), \(d_{ijk} := a_{ijk} + b_{ijk} + c_{ijk} \) \((1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq l)\); the vector \(\omega^T = (x^T, y^T, z^T)\), \(\omega \in \Omega = X \times Y \times Z \subset \mathbb{E}^{m+n+l}\). Next, for \(p = m, n, l\), we define the vectors \(o_p = (0, \ldots, 0) \in \mathbb{E}^p\), \(e_p = (1, \ldots, 1)^T \in \mathbb{E}^p\), as well as \(\mathbb{E}^+_p\) — the nonnegative orthant of the Euclidean space \(\mathbb{E}^p\). The symbol \(^T\) denotes the operation of transposition of a vector (matrix).

Introduce the Nash function (indicator): \(F(\omega) = \delta_x(\omega) + \delta_y(\omega) + \delta_z(\omega)\), where

\[ \delta_x(\omega) = \max_{x' \in X} f_x(x', y, z) - f_x(\omega), \]

\[ \delta_y(\omega) = \max_{y' \in Y} f_y(x, y', z) - f_y(\omega), \]

\[ \delta_z(\omega) = \max_{z' \in Z} f_z(x, y, z') - f_z(\omega). \]

The function \(F(\omega)\) is an analogue of the Nash function defined for the bi-matrix games \([5]\). As the above-defined payoff functions are linear with respect to each variable \(x, y, z\) (when the other two variables are fixed), the game \(\Gamma\) is convex, hence the set of Nash points \(\Omega^*\) is non-empty (but not necessarily convex). For the definition of the convex games and their properties the reader is referred to \([6]\).

Since \(F(\omega) \geq 0\) for all \(\omega \in \Omega\), and \(F(\omega) = 0\) if, and only if \(\omega\) is the (equilibrium) point of the game \(\Gamma\), one can find the solution of game \(\Gamma\) as the global minimum (equalling zero) of the function \(F(\omega)\) on \(\Omega\).

**The 3LP-Method for Solving the 3-Persons Game in the General Setting**

Set the iteration counter \(t = 0\). As an initial (starting) strategy, one can use any pair of the players’ pure strategies (the total number of such pairs is \(mn + ml + nl\)); for example, fix a pair of strategies \(\{y^{(0)}, z^{(0)}\}\) with the components \(y_1^{(0)} = 1, y_j^{(0)} = 0, j = 2, \ldots, n, z_1^{(0)} = 1, z_k^{(0)} = 0, k = 2, \ldots, l, \) and solve successively (for \(t = 0, 1, \ldots\) ) the triple problem \(P_x(y^{(t)}, z^{(t)}; x^{(t+1)})\),
\( P_y(x(t+1), z(t); y(t+1); x(t)) \), and \( P_z(x(t+1), y(t+1); z(t+1)); y(t) \), where

\[
P_x(y', z'; x) : \sum_{i=1}^{m} \left( \sum_{j=1}^{n} \sum_{k=1}^{l} d_{ijk} y'_j z'_k \right) x_i - \beta - \gamma \rightarrow \max_{x, \beta, \gamma},
\]

\[
\sum_{i=1}^{m} \left( \sum_{j=1}^{n} b_{ijk} z'_k \right) x_i - \beta \leq 0, \quad j = 1, \ldots, n,
\]

\[
\sum_{i=1}^{m} \left( \sum_{j=1}^{n} c_{ijk} y'_j \right) x_i - \gamma \leq 0, \quad k = 1, \ldots, l,
\]

\[
x^T e_m = 1, \quad x \succeq o_m, \quad \beta, \gamma \in \mathbb{E}_+^l.
\]

If \( x^* \) is an optimal solution of this problem, then we set \( x' := x^* \); next we solve:

\[
P_y(x', z'; y) : \sum_{j=1}^{n} \left( \sum_{i=1}^{m} \sum_{k=1}^{l} d_{ijk} x'_i z'_k \right) y_j - \alpha - \gamma \rightarrow \max_{y, \alpha, \gamma},
\]

\[
\sum_{j=1}^{n} \left( \sum_{k=1}^{l} a_{ijk} z'_k \right) y_j - \alpha \leq 0, \quad i = 1, \ldots, m,
\]

\[
\sum_{j=1}^{n} \left( \sum_{i=1}^{m} c_{ijk} y'_j \right) y_j - \gamma \leq 0, \quad k = 1, \ldots, l,
\]

\[
y^T e_n = 1, \quad y \succeq o_n, \quad \alpha, \gamma \in \mathbb{E}_+^l.
\]

Again, if \( y^* \) is an optimal plan for the above problem, then put \( y' := y^* \), and continue solving:

\[
P_z(x', y'; z) : \sum_{k=1}^{l} \left( \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ijk} x'_i y'_j \right) z_k - \alpha - \beta \rightarrow \max_{z, \alpha, \beta},
\]

\[
\sum_{k=1}^{l} \left( \sum_{i=1}^{m} b_{ijk} x'_i \right) z_k - \alpha \leq 0, \quad j = 1, \ldots, n,
\]

\[
\sum_{k=1}^{l} \left( \sum_{j=1}^{n} c_{ijk} y'_j \right) z_k - \beta \leq 0, \quad i = 1, \ldots, l,
\]

\[
z^T e_l = 1, \quad z \succeq o_l, \quad \alpha, \beta \in \mathbb{E}_+^l.
\]

Now that \( z^* \) is an optimal solution of that problem, we denote \( z' := z^* \).

The optimal objective function values \( F_t \equiv F(\omega(t+1)) \) are monotone non-increasing by \( t \). The iteration process continues until the value \( F_t \) stabilizes, that is, for some \( t^* \), the difference \( F_{t^*} - F_{t^*+1} \) becomes small enough. In addition, if \( F_{t^*} = 0 \), it means that an (exact) Nash point has been found. If the value \( F_{t^*} \) is positive but small enough, an approximate solution of the game is reported. Otherwise, select a new pair of the initial strategies and start the process again (probably, having altered the order of the solved problems \( P_x, P_y, P_z \)).

**The 3-Persons Game in a Special Setting: the Hexamatrix Game**

We propose this special setting of the 3-persons game in order to check the efficiency of the 3LP-method as applied to such particular cases, as well as to develop a new algorithm inapplicable for the games in the general form but quite reliable when solving the special games.
Consider the set of hexamatrix games (i.e., the poly-matrix 3-persons games) introduced by A.S. Strekalovsky [7]. A hexamatrix game is defined by 6 matrices:

\[ A_1 = (a_{ij}), \quad A_2 = (a_{ik}), \quad B_1 = (b_{jk}), \quad B_2 = (b_{jk}), \quad C_1 = (c_{ki}), \quad C_2 = (c_{kj}), \]

with the players’ payoff functions of the form

\[ f_x(\omega) = x^T(A_1 y + A_2 z), \quad f_y(\omega) = y^T(B_1^T x + B_2 z), \quad f_z(\omega) = z^T(C_1^T x + C_2^T y). \]

The **LH-Method** to find a Nash point.

However, in the case of the 3-persons game in the special setting such a transformation

\[ x \rightarrow \frac{x}{\alpha}, \quad y \rightarrow \frac{y}{\beta}, \quad z \rightarrow \frac{z}{\gamma} \]

The LH-algorithm first applies the linear transformation of variables in the linear complementarity problem associated with the game in question.

The 3LP-Algorithm Solving Hexamatrix Games

In order to solve a hexamatrix game \( \Gamma \) one needs to solve the quadratic programming problem:

\[
P: \quad \begin{align*}
x^T(A_1 y + A_2 z) + y^T(B_1^T x + B_2 z) + z^T(C_1^T x + C_2^T y) - \alpha - \beta - \gamma & \rightarrow \max, \\
A_1 y + A_2 z & \leq \alpha e_m, \\
B_1^T x + B_2 z & \leq \beta e_n, \\
C_1^T x + C_2^T y & \leq \gamma e_l, \\
x^T e_m = 1, \quad x \geq o_m, \\
y^T e_n = 1, \quad y \geq o_n, \\
z^T e_l = 1, \quad z \geq o_l, \quad \alpha, \beta, \gamma \in E_+.
\end{align*}
\]

The hexamatrix game can be solved by the same 3LP-method if one replaces the above-listed formulas \( P_x(y', z'; x) \), \( P_y(x', z'; y) \) and \( P_z(x', y'; z') \) with the following ones:

\[
P_x(y', z'; x): \quad \begin{align*}
x^T(A_1 y' + A_2 z') + y^T(B_1^T x' + C_1^T y') + (y')^T B_2 z' + (z')^T C_2^T y' - \beta - \gamma & \rightarrow \max, \\
B_1^T x & \leq \beta e_n - B_2 z', \\
C_1^T x & \leq \gamma e_l - C_2^T y', \\
x^T e_m = 1, \quad x \geq o_m, \quad \beta, \gamma \in E_+,
\end{align*}
\]

\[
P_y(x', z'; y): \quad \begin{align*}
y^T(B_1^T x' + B_2 z' + A_1^T x' + C_1^T z') + (x')^T A_2 z' + (z')^T C_1^T x' - \alpha - \gamma & \rightarrow \max, \\
A_1 y & \leq \alpha e_m - A_2 z', \\
C_1^T y & \leq \gamma e_l - C_1^T x', \\
y^T e_n = l, \quad y \geq o_n, \quad \alpha, \gamma \in E_+,
\end{align*}
\]

\[
P_z(x', y'; z): \quad \begin{align*}
z^T(C_1^T x' + C_2^T y' + A_2 x' + B_2 y') + (x')^T A_1 y' + (y')^T B_1^T x' - \alpha - \beta & \rightarrow \max, \\
B_2 z & \leq \beta e_n - B_1^T x', \\
A_2 z & \leq \alpha e_m - A_1 y', \\
z^T e_l = 1, \quad z \geq o_l, \quad \alpha, \beta \in E_+.
\end{align*}
\]

The LH-Algorithm Solving the Hexamatrix Game

The Lemke–Howson (LH) algorithm finds the solution of a game by solving the system of linear constraints in the linear complementarity problem associated with the game in question. The LH-method has been described in the paper [3], in which we compared the efficiency of this algorithm with the 2LP-method of finding Nash points of bimatrix games. In more detail, in the case of a bi-matrix game, the LH-algorithm first applies the linear transformation of variables \( x' = x/\alpha, \quad y' = y/\beta \) to obtain the linear complementarity problem tantamount to the original problem. However, in the case of the 3-persons game in the special setting such a transformation fails. Nevertheless, we can generate a series of auxiliary problems (see below) and solve them by the LH-method to find a Nash point.

As the 3-persons game \( Q \) in the special setting stated below

\[
Q: \quad A_1 y + A_2 z + u = \alpha e_m, \quad B_1^T x + B_2 z + v = \beta e_n, \quad C_1^T x + C_2^T y + w = \gamma e_l,
\]

\[
x^T e_m = 1, \quad y^T e_n = 1, \quad z^T e_l = 1, \\
x, u \in E_+^m, \quad y, v \in E_+^n, \quad z, w \in E_+^l, \quad \alpha, \beta, \gamma \geq 0,
\]

268
has all linear constraints, one is inclined to try to solve it with the Lemke–Howson algorithm as a bi-matrix game. However, in contrast to a bi-matrix game, here we have to solve an auxiliary problem in order to generate a starting point.

Namely, the following complementarity conditions must be valid at any Nash equilibrium point:

\[ x_i u_i = 0, \quad 1 \leq i \leq m, \quad y_j v_j = 0, \quad 1 \leq j \leq n, \quad z_k w_k = 0, \quad 1 \leq k \leq l. \]

Having set \( \alpha = 1, \beta = 1, \gamma = 1 \), we reduce the auxiliary problem to the equations system

\[
\begin{align*}
A_1 y + A_2 z + u &= e_m, \\
B_1 x + B_2 z + v &= e_n, \\
C_1 x + C_2 y + w &= e_l,
\end{align*}
\]

\[
x, u \in E^m_+; \quad y, v \in E^n_+; \quad z, w \in E^l_+.
\]

Now introduce the notation:

\[
q = m + n + l,
\]

\[
H = \begin{pmatrix}
O_m & A_1 & A_2 \\
B_1 & O_n & B_2 \\
C_1 & C_2 & O_l
\end{pmatrix}, \quad s = \begin{pmatrix}
x \\
y \\
z
\end{pmatrix}, \quad \sigma = \begin{pmatrix}
u \\
v \\
w
\end{pmatrix}, \quad e = e_q, \quad o = o_q, \quad E = \text{diag}(e),
\]

where for \( p = m, n, l \), all elements of the square matrix \( O_p \) of dimension \( p \) are zero, \( E \) is the unit matrix of dimension \( q \), we generate the following complementarity problem: Find a nonnegative solution \( (s, \sigma \geq 0) \) to the linear system \( H s + E \sigma = e \). In order to solve the latter problem, make use of a procedure of Lemke–Howson type. Namely, select \( \sigma \) as an initial basis. Introduce the variable \( x_1 \) into the basis. As a result, another variable leaves the basis without breaking the nonnegativity of the basic variables. Next, the variable related to the leaving basic variable in the complementarity condition is introduced into the basis. This operation is repeated until the variable \( x_1 \) leaves the basis. The result will be a “pseudo-Nash” point. As the Lemke–Howson procedure may suffer looping, we impose an additional restriction: after starting from a current initial point, we admit only a fixed (bounded) number of iterations, e.g., not exceeding \( q \) (which is the game’s dimension). Here, the term “Nash pseudo-equilibrium” is cited in quotes, because when solving the auxiliary system we relaxed (didn’t include) the restrictions \( x^T e_m = 1, y^T e_n = 1, z^T e_l = 1 \).

The starting points are selected successively, in the order of the components of the vector \( s \): \( x_1, \ldots, x_m, y_1, \ldots, y_n, z_1, \ldots, z_l \) (\( n + m + l \) in total). The thus generated basis is used as an initial basis for the problem \( Q \). The latter is constructed in the following manner. First, we introduce into the basis the basic variables of the auxiliary problem and put them into the same positions that they occupied in the basis of the auxiliary problem. Sure there will be some variables that are impossible to introduce into the basis in their own positions because the leading coefficients of the decompositions of the corresponding columns with the respect to the current basis might be very close to zero. Those variables are introduced into the basis in the yet occupied positions but such that the leading coefficients have large enough absolute values. The solution thus generated may contain negative basic values. In the latter case, we repeat the solution of the auxiliary problem starting from the next initial point. The obtained solution with nonnegative basic values has the same structure as the solution of the auxiliary problem, that is, it satisfies the complementarity conditions, and hence is a Nash point.

The conducted tests demonstrated that a relatively moderate number of “pseudo-Nash” points was necessary to generate before obtaining the final solution.
Test Results for the 3LP- and LH-Algorithms

We tested the algorithms solving the 3-persons games in both the general and the special settings by making use of the personal computer with the processor Intel(R) Core(TM) i5-3427U CPU @ 1.80GHz 2.300 GHz, memory 4.00 GB, 4 cores). The test codes were written in the languages FORTRAN, MatLab, and Python. A series of $S$ games was solved for each triple $n,m,l$.

The matrices were generated in two stages. For the games in the general setting, we first used a pseudo-random counters to generate independently the elements of the auxiliary tables $a'_{ijk},b'_{ijk},c'_{ijk}$ ($1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq l$); at the second stage, we constructed the payoff tables by the formulas

$$a_{ijk} = a'_{ijk} + \kappa(b'_{ijk} + c'_{ijk}), \quad b_{ijk} = b'_{ijk} + \kappa(a'_{ijk} + c'_{ijk}), \quad c_{ijk} = c'_{ijk} + \kappa(a'_{ijk} + b'_{ijk})$$

for all $i,j,k$. Here $\kappa$ is the coefficient of reciprocal dependence of the payoff tables, $0 \leq \kappa \leq 1/2$.

For the games in the special setting, this procedure can be made only when $n = m = l$. In the latter case, we first generated independently the pseudo-random matrices $A_1', A_2', B_1', B_2', C_1', C_2'$, then produced the payoff matrices as follows:

$$A_i = A_i' + \kappa(B_i' + C_i'), \quad B_i = B_i' + \kappa(A_i' + C_i'), \quad C_i = C_i' + \kappa(A_i' + B_i'), \quad i = 1, 2.$$  

If the sizes $n,m,l$ are the same, then the games generated in those two settings are different and are solved differently, too. In the case of the general setting, one can solve games up to the dimension $n = m = l = 350$, while in the case of the special setting — up to 2000. However, the 3LP-method may fail by requiring an unacceptable computational time.

The upper part of Table 1 reports the results of the LH-algorithm solving the set of test games (5 series with 10 instances in each) in the general setting with independent matrices ($\kappa = 0$). The algorithm switched to the next initial pair of strategies after having made $dim$ iterations.

In the lower part of Table 1, the initial pair of strategies were changed after $dim/4$ iterations passed by the LH-algorithm. The two bottom lines of the table report the results of solving a game of dimensions $n = m = l = 500$ and with dependent payoff matrices (the reciprocal dependence coefficient ($\kappa$) equals 0.1 and 0.2, respectively).

Here in Table 1 and onward, the following notation is used: $n,m,l$ are the game’s sizes, $dim = n + m + l$ — the game’s dimension; $S$ — the number of games in each series; $\kappa$ — the coefficient of the reciprocal dependence of the payoff matrices; $IP$ — the total number of the used initial starting strategies; $LP$ — the total number of the simplex-type steps; Time — the total amount of time to solve $S$ games (sec); $neg$ — the total number of Nash pseudo-equilibria with negative basic values in the basis generated for the original problem/game.

Is easy to see from the reported results (see the last three rows in table 1.), the reciprocal dependence of the payoff matrices doesn’t affect much or even decreases the computational efforts to solve a problem in the special setting by the LH-algorithm. However, for the general setting and the 3LP-algorithm, the situation was different. The reciprocal dependence sufficiently increases the complexity of problems. Such results were produced and reported in detail in our previous paper [2]. For example, on the one hand, when solving the series of 10 problems 100x100x100 with independent matrices the result was yielded in 276 sec by making use of 48 initial points. On the other hand, when the matrices were reciprocally dependent ($\kappa = 0.1$) we had to use 15440 starting points in order to finish in 78816 sec.
Table 1. The LH-algorithm solving the family of 50 three-person games in the special setting

<table>
<thead>
<tr>
<th></th>
<th>n = m = l</th>
<th>S</th>
<th>(\kappa)</th>
<th>IP</th>
<th>LP</th>
<th>Time</th>
<th>neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>dim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>10</td>
<td>0.0</td>
<td>101</td>
<td>5812</td>
<td>123</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>10</td>
<td>0.0</td>
<td>261</td>
<td>21732</td>
<td>1155</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>10</td>
<td>0.0</td>
<td>551</td>
<td>77394</td>
<td>5188</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>10</td>
<td>0.0</td>
<td>902</td>
<td>131059</td>
<td>21428</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>10</td>
<td>0.0</td>
<td>1456</td>
<td>120947</td>
<td>57013</td>
<td>28</td>
</tr>
<tr>
<td>dim/4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>10</td>
<td>0.0</td>
<td>447</td>
<td>7408</td>
<td>177</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>10</td>
<td>0.0</td>
<td>668</td>
<td>16510</td>
<td>864</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>10</td>
<td>0.0</td>
<td>1074</td>
<td>35424</td>
<td>3261</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>10</td>
<td>0.0</td>
<td>1144</td>
<td>77157</td>
<td>6020</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>10</td>
<td>0.0</td>
<td>2464</td>
<td>104020</td>
<td>27015</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>10</td>
<td>0.1</td>
<td>756</td>
<td>44334</td>
<td>8394</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>10</td>
<td>0.2</td>
<td>1406</td>
<td>41064</td>
<td>14195</td>
<td>28</td>
</tr>
</tbody>
</table>

Table 2. The results of solving the game \(100 \times 100 \times 100\) in the special s setting by the LH- and 3LP-algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>S</th>
<th>(\kappa)</th>
<th>IP</th>
<th>LP</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python, LH</td>
<td>10</td>
<td>0.0</td>
<td>148</td>
<td>7330</td>
<td>114</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.1</td>
<td>163</td>
<td>8072</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.2</td>
<td>114</td>
<td>5456</td>
<td>46</td>
</tr>
<tr>
<td>Matlab, 3LP</td>
<td>10</td>
<td>0.0</td>
<td>10043</td>
<td>10339212</td>
<td>48893</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.1</td>
<td>6863</td>
<td>6717492</td>
<td>28450</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.2</td>
<td>6135</td>
<td>5894160</td>
<td>27471</td>
</tr>
<tr>
<td>Python, 3LP</td>
<td>5</td>
<td>0.0</td>
<td>3000</td>
<td>1166130</td>
<td>17813</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.1</td>
<td>1641</td>
<td>666629</td>
<td>10946</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.2</td>
<td>2657</td>
<td>1064017</td>
<td>15169</td>
</tr>
</tbody>
</table>

Conclusions

The realized theoretical research and numerical experiments allow one to evaluate the computational efficiency of the 3LP- and LH-algorithms, as well as to discover their strong and minor points.

We have successfully adapted the Lemke–Howson method to the solution of 3-persons games in the special setting: hexamatrix games. All the tested examples have been solved by making use of essentially smaller numbers of initial strategies and much lower computational time than the 3LP-algorithm.

References


