Les Cahiers du GERAD ISSN: 0711–2440

The XGame Solver Software: Equilibria Enumeration and Refinement in Game Theory

S. Belhaiza

G–2010–35

mai 2010

The XGame Solver Software: Equilibria Enumeration and Refinement in Game Theory

Slim Belhaiza

GERAD & École Polytechnique de Montréal
and King Fahd University of Petroleum and Minerals
Department of Mathematics and Statistics
Dhahran, 31261, Saudi Arabia
slimb@kfupm.edu.sa

mai 2010

Les Cahiers du GERAD

G–2010–35

Abstract

In this paper we present the implementation of the EEE and $E\chi MIP$ algorithms with exact arithmetics in a user friendly application with interfaces named *XGame Solver*, using both QT and $C++$ languages. We describe how this software can be used to solve bimatrix games, polymatrix games and sequential forms of two-persons extensive games.

Key Words: XGame, enumeration, bimatrix game, polymatrix, sequential, extensive.

Résumé

Dans ce travail nous présentons l'implantation des deux algorithmes EEE et $E\chi MIP$ dans une application conviviale avec des interfaces dans le cadre d'une application nommée *XGame Solver*, en utilisant les langages QT et $C++$. Nous décrivons comment ce logiciel peut être utilisé afin de résoudre des jeux bimatriciels, polymatrciels et la forme séquentielle de jeux étendus à deux joueurs.

Mots clés : XGame, énumération, jeu bimatriciel, polymatrciel, séquentiel, étendu.

Acknowledgments: Author would like to thank Alexandre Dzimi Mve for his contributions to the development of the XGame Solver Software.

1 Introduction

Different formulations were proposed in order to solve Bimatrix games, Polymatrix games or Sequential forms of two persons extennsive games. While most of the papers studied these games just by defining methods or algorithms to find a Nash equilibrium, few papers used an enumeration approach in order to obtain a complete view of the solutions sets. We used mathematical programming in order to enumerate all Nash extreme equilibria of Bimatrix games, Polymatrix games and Sequential forms of two persons extensive games. While the enumeration algorithm EEE [1] was esentially based on bilevel programming and enumerated bimatrix games extreme Nash equilibria, our work consisted in modeling games as $0-1$ mixed linear programs by linearizing complementarity conditions and introducing well defined parameters. Based on this different approach, $E\chi Mip$ algorithm [2, 4] was proposed and implemented to enumerate extreme Nash equilibria of bimatrix games, three persons Polymatrix games and Sequential forms of two persons extensive games. Both of the algorithms were first implemented using Cplex callable libraries. This state of art software helped increasing the size of the problems that could be solved to 29×29 for bimatrix games.

In order to improve the precision and the independance of our algorithms we reimplemented these algorithms using exact arithmetics libraries. We also implemented a maximal cliques enumeration algorithm and proposed some automatic refinements procedures of all extreme Nash equilibria for Bimatrix games in [3, 5]. To make the scientific community benefit from our work, we gathered our latest implementations of our algrithms and refinements procedures into a Software package called *XGame Solver*.

This papers is organized as follows. In the first section we recall the different formulations we have used in order to enumerate extreme Nash equilibria of bimatrix games,Polymatrix games and Sequential forms of two persons extensive games. The second section presents the pseudo-codes of the EEE and $E\chi Mip$ algorithms adapated to these games. Section three describes the *XGame Solver* software and the different icons and options it provides to the user.

2 Formulations

This section recalls the different formulations we proposed in order enumerate all extreme Nash equilibria for bimatrix games, three persons Polymatrix games and Sequential forms of two persons extensive games.

2.1 Bimatrix Games

A bimatrix game, or two-person nonzero-sum game in strategic form, may be stated as follows: given a pair of $m \times n$ payoff matrices A and B, Player I attempts to maximize his payoff $x^t A y$ by selecting the probability vector x in $\mathbb{R}^m$, and simultaneously, Player II attempts to maximize his payoff $x^t B y$ by choosing the probability vector y in $\mathbb{R}^n$.

It is well known that there is always at least one *equilibrium point* (Nash [11]) in such a game, i.e., a pair of strategies $(\hat{x}, \hat{y})$ that simultaneously solve the two parameterized linear programs

$$(P1) \quad \begin{array}{ll} \max\limits_{x} & x^t A\hat{y} \\ \text{s.t.} & x^t\mathbf{1} = 1, \\ & x \geq \mathbf{0}, \end{array} \qquad \text{and} \qquad (P2) \quad \begin{array}{ll} \max\limits_{y} & \hat{x}^t B y \\ \text{s.t.} & \mathbf{1}^t y = 1, \\ & y \geq \mathbf{0}, \end{array}$$

where $\mathbf{0}$ and $\mathbf{1}$ respectively denote vectors whose components are all equal to zero and one. The strategy $\hat{x}$ is a best response to $\hat{y}$, as it is an optimal solution of (P1); and $\hat{y}$ is a best response to $\hat{x}$, as it is an optimal solution of (P2). At an equilibrium point neither player has an incentive to change his strategy unless the other player does so.

Linear programming duality theory yields the following equivalent dual programs

$$(D1) \quad \begin{array}{ll} \min\limits_{\alpha} & \alpha \\ \text{s.t.} & \mathbf{1}\alpha \geq A\hat{y}, \end{array} \qquad \text{and} \qquad (D2) \quad \begin{array}{ll} \min\limits_{\beta} & \beta \\ \text{s.t.} & \beta\mathbf{1}^t \geq \hat{x}^t B. \end{array}$$

where α and β are dual variables in $\mathbb{R}$.

There are two classical and equivalent necessary and sufficient optimality conditions in linear programming theory. In our framework, these conditions are expressed as the equilibrium conditions. Both conditions require primal and dual feasibility: the pair of strategies $(\hat{x}, \hat{y})$ together with scalars $\hat{\alpha}$ and $\hat{\beta}$ must satisfy

$$\begin{aligned}(\hat{x}, \hat{\beta}) \in X &= \{(x, \beta) \in \mathbb{R}^{m+1} : \ x^t B \leq \beta \mathbf{1}^t, \ x^t \mathbf{1} = 1, \ x \geq \mathbf{0}\}, \\ (\hat{y}, \hat{\alpha}) \in Y &= \{(y, \alpha) \in \mathbb{R}^{n+1} : \ Ay \leq \mathbf{1}\alpha, \ \mathbf{1}^t y = 1, \ y \geq \mathbf{0}\}.\end{aligned}$$

The first optimality condition consists in the equality of primal and dual objective function values

$$\hat{x}^t A \hat{y} = \hat{\alpha} \qquad \text{and} \qquad \hat{x}^t B \hat{y} = \hat{\beta}. \tag{1}$$

Which means that the values of the dual variables must be equal to the payoffs. The second optimality condition is the complementary slackness condition

$$\hat{x}^t (\mathbf{1}\hat{\alpha} - A\hat{y}) = 0 \qquad \text{and} \qquad (\hat{\beta}\mathbf{1}^t - \hat{x}^t B)\hat{y} = 0. \tag{2}$$

Equivalence of optimality conditions (1) and (2) appears when substituting $\hat{x}^t \mathbf{1}$ and $\mathbf{1}^t \hat{y}$ by 1.

There may be one, many, or an infinity of equilibrium points. Indeed, for a given strategy $\hat{y}$, the set of optimal responses of Player I, i.e., the set of optimal solutions of (P1), denoted $X(\hat{y})$, is either a singleton or a polytope. The symmetric observation holds for (P2), where the set of optimal solutions is denoted $Y(\hat{x})$. In fact, the set E of all equilibrium points is the union of a finite number of polytopes called *maximal Nash subsets* [10]. We have shown [3] that all maximal Nash subsets can be enumerated using a maximal cliques enumeration algorithm.

2.1.1 *EEE* on bimatrix games

With *EEE* algorithm Enumeration of all extreme equilibria is achieved through exploration of a search tree. To each node of the tree correspond two linear subprograms whose feasible regions are derived from X and Y. They differ through conversion of inequalities into equalities. Two types of branching rules define these conversions.

The first type of rule relies on an equivalent formulation of the complementary slackness condition (2). At an equilibrium point, the feasibility conditions insure that the vectors $\hat{x}$ and $\hat{y}$ are nonnegative, and that $\hat{x}^t B \leq \hat{\beta}\mathbf{1}^t$ and $A\hat{y} \leq \mathbf{1}\hat{\alpha}$. Therefore, condition (2) can be written through the $m+n$ complementary slackness conditions

$$\hat{x}_i = 0 \ \text{ or } \ A_{i\cdot}\hat{y} = \hat{\alpha} \qquad \text{and} \qquad \hat{y}_j = 0 \ \text{ or } \ \hat{x}^t B_{\cdot j} = \hat{\beta}, \tag{3}$$

where $A_{i\cdot}$ denotes the i^{th} row of A, and $B_{\cdot j}$ the j^{th} column of B for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. The first type of branching rule splits the current node of the tree through two branches: in one branch, a pure strategy is forced to be played with probability zero (a variable is fixed to zero), and in the second, a pure strategy is forced to have maximum payoff, so it is a best response a slack variable is fixed to zero). This rule is invoked by the algorithm until all complementary slackness conditions (3) are forced to be satisfied, and thus for all nodes whose depth in the search tree is less than or equal to $m + n$.

When the depth of the current node exceeds $m + n$, the second type of branching rule is used. At such a node, all complementary slackness conditions (3) are guaranteed to be satisfied by the first type of branching rule, and thus any solution of the current converted equalities is necessarily an equilibrium point. This branching rule consists in splitting the current node through as many branches as there are strictly positive variables and slack variables of X and Y corresponding to the equilibrium point. In each branch the variable or slack variable is fixed to zero. All degenerate solutions are reached, possibly several times, and thus elimination of duplicates is required.

Backtracking occurs when conversion of inequalities into equalities reduces the feasible region to the empty set. Let us now introduce more formal definitions in order to fully describe the algorithm.

As discussed above, every new branch is obtained by converting a single inequality of X or Y to an equality. In order to efficiently select which inequality to take, we introduce the following linear programming problems, parameterized in the objective functions:

$$\begin{aligned} P(y) &\equiv \max_{(x,\beta)\in X} x^t Ay - \beta, \\ Q(x) &\equiv \max_{(y,\alpha)\in Y} x^t By - \alpha. \end{aligned}$$

Problem $P(y)$ is an aggregation of the primal and dual problems (P1) and (D2), and $Q(x)$ is an aggregation of (P2) and (D1). Indeed, the constraints appearing in the definition of X and Y are those of the primal and dual problems. Moreover, the objective functions $x^tAy - \beta$ and $x^tBy - \alpha$ are the sum of the primal and dual objective functions. Therefore, it is likely that an optimal solution of $P(y)$ solves (P1) and (D2), and one of $Q(x)$ solves (P2) and (D1). The search for equilibria is done through the feasible regions X and Y, and thus any objective functions could be used. The intuitive motivation behind this choice of objective functions, is to guide the algorithm toward equilibrium points.

At each node of the search tree is associated a pair of current subproblems $\tilde{P}$ and $\tilde{Q}$, that are identical to P and Q, except that some of the inequality constraints or nonnegativity constraints are converted to equalities. The depth of the root node is 1. At each node, one of the three following cases occurs:

i- either $\tilde{P}$ or $\tilde{Q}$ is infeasible;
ii- both $\tilde{P}$ and $\tilde{Q}$ are feasible but there is not enough information to obtain an equilibrium point (depth of the current node is less than or equal to $m + n$);
iii- both $\tilde{P}$ and $\tilde{Q}$ are feasible and there is sufficient information to deduce an extreme equilibrium point (depth of the current node is greater than $m + n$).

In the first case, the current node is discarded. In the two other cases, the current node is split into others through new branches. The subproblems of the new nodes are identical to the current ones except for one additional constraint converted into an equality in either $\tilde{P}$ or $\tilde{Q}$, chosen according to the branching rules. Thus for each new branch, only one of the two subproblems needs to be checked for feasibility.

In the first type of branching rule, the selected pair of complementary inequalities to be converted into equalities corresponds to the largest complementary product $x_i(\alpha - A_{i\cdot}y)$ or $(\beta - x^tB_{\cdot j})y_j$, where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. Of course, the pair of inequalities must be chosen among those that were not already converted in $\tilde{P}$ or $\tilde{Q}$.

A node is not only characterized by the pair of current subproblems $\tilde{P}$ and $\tilde{Q}$. One of the two vectors x or y is also associated to each node. This vector describes a feasible strategy for the current subproblem that does not differ from its predecessor in the tree. We now introduce the notation that specifies which constraint is converted. This notation will be used when formally defining both types of branching rule. Given the current subproblems $\tilde{P}$ and $\tilde{Q}$, we define:

P^i to be $\tilde{P}$ in which the constraint $x_i \geq 0$ is converted to $x_i = 0$,
P_j to be $\tilde{P}$ in which the constraint $x^tB_{\cdot j} \leq \beta$ is converted to $x^tB_{\cdot j} = \beta$,
Q^j to be $\tilde{Q}$ in which the constraint $y_j \geq 0$ is converted to $y_j = 0$,
Q_i to be $\tilde{Q}$ in which the constraint $A_{i\cdot}y \leq \alpha$ is converted to $A_{i\cdot}y = \alpha$,

where $i \in \{1, 2, \ldots, m\}$ and $j \in \{1, 2, \ldots, n\}$.

2.1.2 $E\chi Mip$ on bimatrix games

Complementary slackness conditions were stated as $x_i(\mathbf{1}\alpha - A_{i\cdot}y) = 0$, and $(\mathbf{1}\beta - x^tB_{\cdot j})y_j = 0$, for $i \in \{1, 2, \ldots, m\}$ and for $j \in \{1, 2, \ldots, n\}$, or in matrix form

$$x^t(\mathbf{1}\alpha - Ay) = 0, \qquad \text{and} \qquad (\beta\mathbf{1}^t - x^tB)y = 0. \tag{4}$$

Pairs of solutions (x, α) or (y, β) feasible for the primal and the dual, and satisfying the complementarity slackness conditions are optimal. If this holds simultaneously for the programs of both players, the solution (x, y) is a Nash equilibrium.

Linearization of these complementary slackness conditions is made possible through the use of $0-1$ variables (Júdice and Mitra [7] and Audet et al. [1])

$$(\mathbf{1}\alpha - Ay) \leq L_1 u, \qquad \text{and} \qquad (\beta \mathbf{1}^t - x^t B) \leq L_2 v, \tag{5}$$

$$x + u \leq \mathbf{1}, \qquad \text{and} \qquad y + v \leq \mathbf{1}, \tag{6}$$

$$u \in \{0,1\}^n, \qquad \text{and} \qquad v \in \{0,1\}^m, \tag{7}$$

where L_1 and L_2 are some large constants. Making sure that the constants L_1 and L_2 are large enough is often problematic. Fortunately, in our case, the following result shows how we can easily obtain some valid values.

$$L_1 = \max_{i,\ j} a_{ij} - \min_{i,\ j} a_{ij}, \qquad L_2 = \max_{i,\ j} b_{ij} - \min_{i,\ j} b_{ij}.$$

The set of bimatrix game Nash equilibria is the set of pairs of mixed strategies $(x, y) \in \mathbb{R}^m \times \mathbb{R}^n$ for which there exists vectors $(u, v) \in \{0,1\}^m \times \{0,1\}^n$, satisfying

$$\begin{array}{ll}
x^t\mathbf{1} = 1, & \mathbf{1}^t y = 1, \\
x^t B - \beta \mathbf{1}^t \ \leq \ 0, & Ay - \ \alpha \mathbf{1} \ \leq \ 0, \\
(\mathbf{1}\alpha - Ay) \ - \ L_1 u \leq \ 0, & (\beta \mathbf{1}^t - x^t B) - \ L_2 v \leq \ 0, \\
x + u \leq \mathbf{1}, & y + v \leq \mathbf{1}, \\
x \geq 0, & y \geq 0, \\
u \in \{0,1\}^m & v \in \{0,1\}^n.
\end{array}$$

Complete enumeration of all bimatrix game extreme equilibria can be done through complete enumeration of all extreme feasible solutions of a mixed $0-1$ linear problem (i.e., extreme feasible solutions for each feasible $0-1$ vector), with any linear objective function.

$E\chi MIP$ is designed to generate a branching tree where at each node a dichotomous branching over one of the binary variables is done. The exploring tree generated contains a principal tree and many secondary trees.

The principal tree is designed to detect all binary variables combinations involved in one or more extreme equilibria. A secondary tree is generated from every principal tree node offering a feasible solution, i.e. an extreme equilibrium. Hence, each binary variable combination involved in an extreme equilibrium is completely explored in order to find all extreme equilibria that could be obtained from it. To avoid repetitive exploration of the same binary variables combination, an eliminating constraint is added once secondary exploration is achieved. These constraints are often redundant and could be eliminated in part to reduce the problem size.

At each node of the principal tree, a mixed $0-1$ linear program is solved. This program is composed of the original program with some binary variables fixing constraints and some combination of binary variables eliminating constraints.

Each secondary tree node represents the solution of a linear problem, composed of the original problem with some binary variable fixing constraints, some combination of binary variables eliminating constraints and some continuous variables fixing constraints.

2.2 Sequential form of two-persons extensive games

The use of *sequences* of moves instead of pure strategies is possible in order to compute equilibria for extensive form games. As the extensive game is represented by a tree, there exists a unique path linking the root node to any node of such a tree. This path defines a sequence of moves for player i. Assuming that each player i has perfect recall means that each pair of nodes in an information set h in H_i correspond to the same sequence for player i.

In most papers, a sequence of moves is denoted σ_h and the set of move sequences is denoted S_i for each player i. Any sequence of moves $\sigma \in S_i$ can either be equal to the empty sequence $\emptyset$ or only given by its last move at the information set $h \in H_i$, which means that $\sigma\sigma_h c$. This leads to the following definition

$$S_i = \{\emptyset\} \cup \{\sigma_h c \mid h \in H_i,\ c \in C_h\}.$$

Following this definition, each player will have a number of sequences that does not exceed the number of nodes in the tree. The *sequence form* of an extensive game is similar to its strategic form reduction. The only difference between these two conversions is that the sequence form uses sequences instead of pure strategies which leads to a more compact description of the original game.

For a given player i, a behavior strategy β is obtained by probabilities $\beta(c)$ for his moves $c \in C_h$ such that $\beta(c) \geq 0$ and $\sum_{c \in C_h} \beta(c) = 1$ for each $h \in H_i$. Behavior strategy's definition can be extended to the sequences $\sigma \in S_i$ simply by the following formulation

$$\beta\left[c\right] = \prod_{c \in \sigma} \beta(c).$$

In this context, a pure strategy π for a given player is a kind of behavior strategy with $\pi(c) \in \{0, 1\}$ for all moves c, which means that $\pi\left[\sigma\right] \in \{0, 1\}$ for each $\sigma \in S_i$. Thus, a mixed strategy μ corresponds to a probability $\mu(\pi)$ to every pure strategy π of a player i. The *realization probabilities* of playing the sequences $\sigma \in S_i$ are defined as follows

$$\mu\left[\sigma\right] = \sum_{\forall \pi} \mu(\pi)\pi\left[\sigma\right].$$

For a player i, a realization plan of μ is then denoted $x(\sigma) = \mu\left[\sigma\right]$ for $\sigma \in S_i$. For a given player i, x_i is the realization plan of a mixed strategy if and only if the following conditions are satisfied

$$\begin{aligned} x(\emptyset) &= 1, \\ \sum_{c \in C_h} x(\sigma_h c) &= x(\sigma_h), h \in H_i, \\ x(\sigma) &\geq 0,\ \forall \sigma \in S_i. \end{aligned}$$

Koller and Megiddo [8] refer to these conditions using realization probabilities of the game tree. Denoting $x_i = (x_\sigma)_{\sigma \in S_i}$, these conditions can be reformulated for each player i as follows

$$x_i \geq 0, \tag{8}$$
$$E_i x_i = e_i, \tag{9}$$

where E_i is a well chosen matrix and $e_i = (1, 0..., 0)^t$, both with $1 + |H_i|$ rows.

Koller and Megiddo [8] show that two mixed strategies μ and μ' of player i are realization equivalent if and only if they have the same realization plan, i.e. $\mu\left[\sigma\right] = \mu'\left[\sigma\right]$ for all $\sigma \in S_i$.

Moreover, Kuhn [9] show that for a player with perfect recall, any mixed strategy is realization equivalent to a behavioral strategy. Furthermore, Romanovskii [12] and von Stengel [13] demonstrate that the equilibria of a two person game in extensive form with perfect recall are simultaneous solutions of the following pair of parametrized linear programs

$$\begin{array}{llcll} \max\limits_{x_1} & x_1^t A x_2 & \text{and} & \max_{x_2} & x_1^t B x_2 \\ \\ s.t. & E_1 x_1 = e_1, & & s.t & E_2 x_2 = e_2, \\ & x_1 \geq 0, & & & x_2 \geq 0. \end{array} \tag{10}$$

Where E_1 and E_2 are matrices with all elements equal to 1, 0 or -1. Each matrix E_1 or E_2, has as many columns as the number of sequences of play and as many lines as the number of linked sequences sets, for the corresponding player. The single columns e_1 and e_2 have the same number of lines as E_1 and E_2, with the first element equal to 1 and all other elements equal to 0.

The dual formulations of linear programs (10) are expressed as follows

$$\begin{array}{lll} \min\limits_{\alpha_1} \quad e_1^t\alpha_1 & \text{and} & \min_{\alpha_2} \quad e_2^t\alpha_2 \\ s.t. \quad E_1^t\alpha \geq Ax_2, & & s.t. \quad \alpha_2^tE_2 \geq x_1^tB. \end{array} \tag{11}$$

The complementarity constraints obtained from (10) and (11) are

$$x_1^t(E_1^t\alpha_1 - Ax_2) = 0 \qquad \text{and} \qquad (\alpha_2^tE_2 - x_1^tB)x_2 = 0. \tag{12}$$

Using these complementarity conditions, von Stengel et al. [14] define an algorithm able to compute normal form perfect equilibria for two-person games.

By introducing two binary vectors u_1 and u_2 as detailed with bimatrix games, with u_1 and u_2 having as many lines as the number of sequences of the corresponding player, the complementarity onditions can be linearized as follows, with (10) and (11) satisfied

$$x_1^t(E_1^t\alpha - Ax_2) = 0 \Leftrightarrow \begin{array}{lcl} x_1 + u_1 & \leq & \mathbb{1}, \\ E_1^t\alpha - Ax_2 & \leq & L_1u_1, \end{array} \tag{13}$$

and

$$(\alpha_2^tE_2 - x^tB)x_2 = 0 \Leftrightarrow \begin{array}{lcl} x_2 + u_2 & \leq & \mathbb{1}, \\ \alpha_2^tE_2 - x_1^tB & \leq & L_2u_2. \end{array} \tag{14}$$

where $\mathbb{1}$ is a vector with all elements equal to one, and L_1 and L_2 are small scalars equal to the difference between the largest and the least element of each payoff matrix, as described in [2]. On one hand, observe that for any player i, if its i^th binary variable is equal to 1 then its i^th continuous variable is equal to 0 and the complementary slackness condition associated is satisfied. On the other hand, if its i^th binary variable is equal to 0 then the complementary slackness condition associated is also satisfied.

We proposed to achieve the enumeration of all extreme Nash equilibria of the sequence form of a two-persons extensive game by enumerating all extreme points of a set Q defined by the following conditions

$$\begin{array}{ll} E_1x_1 = e_1, & E_2x_2 = e_2, \\ E_1^t\alpha_1 \geq Ax_2, & \alpha_2^tE_2 \geq x_1^tB, \\ x_1 + u_1 \leq \mathbb{1}, & x_2 + u_2 \leq \mathbb{1}, \\ E_1^t\alpha - Ax_2 \leq L_1u_1, & \alpha_2^tE_2 - x_1^tB \leq L_2u_2, \\ x_1 \geq 0, & x_2 \geq 0, \\ u_1 \in \{0,1\} & u_2 \in \{0,1\}. \end{array} \tag{15}$$

Let n_1 and n_2 be the number of sequences of players 1 and 2, respectively. It follows that each binary combination of u_1 and u_2 defines a polytope and that Q is the set of all these disjoint polytopes.

2.3 Polymatrix Games

The confrontation of n players ($n \geq 2$) in a normal and noncooperative context is a *polymatrix game* if payoffs are sums of values for each player and all other ones pairwise. Let now $N = \{1, ..., n\}$ be the set of all players and each player $i \in N$ have a finite set of pure strategies $S_i = \{s_i^1, ..., s_i^{m_i}\}$ with $|S_i| = m_i$.

If player i chooses his strategy s_i^k and player j chooses his strategy s_j^l, a partial payoff $a_{ij}(s_i^k, s_j^l)$ is assigned for player i. For any pure strategic choice $(s_1^k, ..., s_n^l)$ of the n players, the overall player i payoff at the end of the game is

$$A_i(s_1^k, ..., s_n^l) = \sum_{j \neq i} a_{ij}(s_i^k, s_j^l).$$

The $m_i \times m_j$ matrix $A_{ij} = (a_{ij}^{kl})$ is defined as player i's partial payoff matrix relative to player j's strategic decisions. Thus, player i's payoff relative to player j's decisions is not correlated with any of the remaining players' choices.

As in bimatrix games, each polymatrix game player i attempts to maximize his own overall payoff by selecting a probability vector X_i over his set of pure strategies. The *mixed strategy* vector X_i is such that $(X_i)^T = (x_i^1, ..., x_i^{m_i})$, where for all $k \in \{1, ..., m_i\}$, x_i^k is the relative frequency, or probability, with which player i plays his strategy $s_i^k \in S_i$. Hence player i's mixed strategies belong to the set

$$\tilde{S}_i = \{X_i : \ e^t X_i = 1 \ , \ X_i \geq 0\}.$$

The overall payoff of player i at the end of a polymatrix game is

$$R_i(X) = (X_i)^T \sum_{j \neq i} A_{ij} X_j = \sum_{j \neq i} \sum_{k=1}^{m_i} \sum_{l=1}^{m_j} a_{ij}^{kl} x_i^k x_j^l \ .$$

A n-tuple $X^* = (X_1^*, ..., X_n^*)$ of mixed strategies is called a $Nash$ equilibrium, in a polymatrix game, if and only if for any other n-tuple $X = (X_1^*, .., X_{i-1}^*, X_i, X_{i+1}^*, .., X_n^*)$ the following inequality is satisfied

$$(X_i^*)^T \sum_{j \neq i} A_{ij} X_j^* \geq (X_i)^T \sum_{j \neq i} A_{ij} X_j^*, \qquad \text{for } i \in N. \tag{16}$$

i.e., player i's payoff relative to all other players is simultaneously maximized for $i \in N$. It again follows from Nash's [11] that a polymatrix game has at least one equilibrium.

For a set of mixed strategies $X_1, ..., X_n$ and for $i \in N$, let

$$\alpha_i = (X_i^*)^T \sum_{j \neq i} A_{ij} X_j. \tag{17}$$

Considering an $(m_i \times 1)$ column e_r^i with its r^{th} element equal to 1 and all other elements equal to 0 and using the fact that inequality (16) holds for all X_i, even for $X_i = e_r^i$ $(r = 1, ..., m_i)$, (17) holds only if

$$\alpha_i e^i \geq \sum_{j \neq i} A_{ij} X_j, \qquad \text{for } i \in N, \tag{18}$$

where e^i is an $(m_i \times 1)$ column with all elements equal to 1. This leads to the statement

$$(X_i^*)^T \alpha_i e \geq (X_i^*)^T \sum_{j \neq i} A_{ij} X_j \quad \Longrightarrow \quad (X_i^*)^T \left(\sum_{j \neq i} A_{ij} X_j^* - \alpha_i e \right) = 0. \tag{19}$$

This last result implies that each α_i, $i \in N$, corresponds to the overall payoff of player i at an equilibrium. The relation (19) is a first complementarity condition.

Similarly, define

$$Y_i = \alpha_i e - \sum_{j \neq i} A_{ij} X_j \quad \text{and} \quad \mu_i = e^T X_i^* - 1, \qquad \text{for } i \in N.$$

Using (17), (18) and (19) with the fact that X_i^* is a probability vector, the following conditions could be stated

$$X_i \geq 0, \ Y_i \geq 0, \ (X_i^*)^T Y_i = 0, \quad \text{for } i \in N, 11) \tag{20}$$

$$\mu_i \geq 0, \ \alpha_i \geq 0, \ \mu_i \alpha_i = 0, \quad \text{for } i \in N. \tag{21}$$

Considering player i's primal multiparametric linear program in a polymatrix game

$$\begin{array}{rl} \max_{X_i} & X_i \sum_{j\neq i} A_{ij}X_j \\ s.t. & e^t_{m_i} X_i = 1, \\ & X_i \geq 0. \end{array} \qquad \text{and player } i\text{'s dual program} \qquad \begin{array}{rl} \min & \alpha_i \\ s.t. & \\ & \alpha_i e^i \geq \sum_{j\neq i} A_{ij}X_j, \end{array}$$

linearization of all complementary slackness conditions (19) can again be achieved using binary variables. This leads to a mixed $0-1$ linear formulation of a polymatrix game.

For $i \in N$, the complementary slackness conditions are written

$$(X_i)^T \left(\alpha_i e^i - \sum_{j=1, j\neq i}^{n} A_{ij}X_j \right) = 0 \qquad \Longleftrightarrow \qquad \begin{array}{l} \alpha_i e^i - \sum\limits_{j=1,\ j\neq i}^{n} A_{ij}X_j - L_iU_i \leq 0, \\ X_i + U_i \leq e. \end{array}$$

Selection of L_i can again be done by simple arithmetic. L_i is of the same order of magnitude that the input:

$$L_i = \sum_{j=1,\ j\neq i}^{n} \Gamma_{ij}, \qquad \text{for } i \in N,$$

where Γ_{ij} is the difference between the largest and the smallest elements of A_{ij}.

The problem of enumerating extreme Nash equilibria for a polymatrix game can then be stated through mixed integer programming. In fact, the set of polymatrix game equilibria is the set of mixed strategies vectors $(X_1, X_2, ..., X_n) \in \mathbb{R}^{m_1} \times \mathbb{R}^{m_2} \times ... \times \mathbb{R}^{m_n}$ such that for $i \in N$

$$\begin{cases} X_i^t e = 1, \\ X_i + U_i \leq e, \\ \alpha_i e^i - \sum\limits_{j=1,\ j\neq i}^{n} A_{ij}X_j \geq 0, \\ \alpha_i e^i - \sum\limits_{j=1,\ j\neq i}^{n} A_{ij}X_j - L_iU_i \leq 0, \\ X_i \geq 0, \\ U_i \in \{0,1\}^{m_i}. \end{cases}$$

Moreover, the use of mixed integer programming allows a flexibility in the selection of an objective function.

3 Algorithms

This section regroups the pseudo-codes of the *EEE* and *EχMip* algorithms used in order to enumerate all the extreme Nash equilibria of bimatrix games, three persons polymatrix games and sequential forms of two-persons extensive games. Complete proofs of effiency of these algorithms were presented in [1, 2, 4].

3.1 *EEE* Algorithm

We adapted the structure of the *EEE* algorithm to both models obtained from bimatrix games and sequential forms of two persons extensive games. This section presents the pseudo-codes of the *EEE* algorithm on these kind of games.

3.1.1 EEE on Bimatrix games

The algorithm *EEE* applied to bimatrix games is now briefly stated.
Step a. Initialization.
Initialize the set of nodes to be explored to $T = \{(x, P, Q)\}$ where x is set to an arbitrary feasible value

(typically $x = (\frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m})$). Go to Step b.
Step b. Selection of a Node.
If the set T of nodes to be explored is empty, then stop. Otherwise, choose and remove a node N from T. Either $N = (x, \tilde{P}, \tilde{Q})$ or $N = (y, \tilde{P}, \tilde{Q})$. In the first case go to Step c; in the second case, go to Step d.
Step c. Direct Feasibility Test ($\tilde{Q}$).
If $\tilde{Q}(x)$ is infeasible then go to Step b; otherwise choose $(\tilde{y}, \tilde{\alpha})$ that solves $\tilde{Q}(x)$ and $(\tilde{x}, \tilde{\beta})$ that solves $\tilde{P}(\tilde{y})$. If the depth of the current node is not greater than $m + n$, go to Step e; otherwise go to Step f.
Step d. Direct Feasibility Test ($\tilde{P}$).
If $\tilde{P}(y)$ is infeasible then go to Step b; otherwise choose $(\tilde{x}, \tilde{\beta})$ that solves $\tilde{P}(y)$ and $(\tilde{y}, \tilde{\alpha})$ that solves $\tilde{Q}(\tilde{x})$. If the depth of the current node is not greater than $m + n$, go to Step e; otherwise go to Step f.
Step e. Dichotomous Branching. Let

$$\tau_i = \begin{cases} \tilde{x}_i(\alpha - A_{i\cdot}\tilde{y}) & \text{if the variable } x_i \text{ of } \tilde{P} \text{ is not forced to be null, and} \\ & \text{the constraint } A_{i\cdot}y \le \alpha \text{ of } \tilde{Q} \text{ is not fixed at equality,} \\ -1 & \text{otherwise,} \end{cases}$$

$$\pi_j = \begin{cases} (\beta - \tilde{x}^t B_{\cdot j})\tilde{y}_j & \text{if the variable } y_j \text{ of } \tilde{Q} \text{ is not forced to be null,} \\ & \text{the constraint } x^t B_{\cdot j} \le \beta \text{ of } \tilde{P} \text{ is not fixed at equality,} \\ -1 & \text{otherwise,} \end{cases}$$

then select the indices ĩ and j̃ that maximize the values τ_i and π_j (ties are broken arbitrarily) for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. In the case where $\tau_{\tilde{\imath}} \ge \pi_{\tilde{\jmath}}$, add

$$(\tilde{y}, P^{\tilde{\imath}}, \tilde{Q}) \text{ and } (\tilde{x}, \tilde{P}, Q_{\tilde{\imath}}),$$

to T and in the case where $\tau_{\tilde{\imath}} < \pi_{\tilde{\jmath}}$, add

$$(\tilde{x}, \tilde{P}, Q^{\tilde{\jmath}}) \text{ and } (\tilde{y}, P_{\tilde{\jmath}}, \tilde{Q}).$$

to T. Go to Step b.
Step f. Polychotomous Branching. The solution $(\tilde{x}, \tilde{y})$ is an equilibrium strategy since the complementary slackness conditions are satisfied. Record it on the list of equilibria if not already there, and add to T all nodes corresponding to a strictly positive variable or slack variable. These nodes are in the four sets

$$\begin{array}{lcl} \{(\tilde{y}, P^i, \tilde{Q}) : \tilde{x}_i > 0\}, & & \{(\tilde{x}, \tilde{P}, Q_i) : A_{i\cdot}\tilde{y} < \alpha\} \\ \{(\tilde{x}, \tilde{P}, Q^j) : \tilde{y}_j > 0\}, & \text{and} & \{(\tilde{y}, P_j, \tilde{Q}) : \tilde{x}^t B_{\cdot j} < \beta\}. \end{array}$$

Go to Step b.

3.1.2 *EEE* Pseudo-Code on Sequential form of two persons extersive game

This section illustrates the new implementation of the *EEE* algorithm based on the sequential form of a two-person extensive game. We propose an implicit enumeration algorithm to determine all extreme equilibria by enumerating all vertices of X_1 and X_2 that satisfy the complementarity conditions $x_1^t(E_1\alpha_1 - Ax_2) = 0$ and $(\alpha_2^t E_2 - x_1^t B)x_2 = 0$, where

$$\begin{array}{lcl} (\hat{x_1}, \hat{\alpha_2}) \in X_1 & = & \{(x_1, \alpha_2) \in \mathbb{R}^{m+1} : \ x_1^t B \le \alpha_2 E_2^t, \ x_1^t \mathbf{1} = 1, \ x_1 \ge \mathbf{0}\}, \\ (\hat{x_2}, \hat{\alpha_1}) \in X_2 & = & \{(x_2, \alpha_1) \in \mathbb{R}^{n+1} : \ Ax_2 \le E_1\alpha_1, \ \mathbf{1}^t x_2 = 1, \ x_2 \ge \mathbf{0}\}. \end{array}$$

Consider the parametrized linear programs:

$$P(x_2) \equiv \max_{(x_1,\alpha_1)\in X_1} x_1^t A x_2 - \alpha_2^t e_2, \qquad \text{and} \qquad Q(x_1) \equiv \max_{(x_2,\alpha_2)\in X_2} x_1^t B x_2 - e_1^t \alpha_1.$$

Indeed, the constraints appearing in the definition of X_1 and X_2 are those of the primal and dual problems. Again, the objective functions $x_1^t A x_2 - \alpha_2^t e_2$ and $x_1^t B x_2 - e_1^t \alpha_1$ are gaps between primal and dual objective functions.

The structure of the algorithm presented for the strategic form almost directly applies. Given the problems $\tilde{P}$ and $\tilde{Q}$, we redefine:

P^i to be $\tilde{P}$ in which the constraint $x_{1i} \geq 0$ is replaced by $x_{1i} = 0$,
P_j to be $\tilde{P}$ in which the constraint $x_{1i}^t B_{\cdot j} \leq \alpha_2^t E_{2 \cdot j}$ is replaced by $x_1^t B_{\cdot j} \ = \alpha_2^t E_{2 \cdot j}$,
Q^j to be $\tilde{Q}$ in which the constraint $x_{2j} \geq 0$ is replaced by $x_{2j} = 0$,
Q_i to be $\tilde{Q}$ in which the constraint $A_{i \cdot} x_2 \leq E_{1i \cdot}\ \alpha_1$ is replaced by $A_{i \cdot} x_2 = E_{1i \cdot}\ \alpha_1$.

We here state the pseudo code of the algorithm EEE applied to Sequential form of two persons extersive game.

a. Initialization. Initialize the set of nodes to be explored to $T = \{(x_1, P, Q)\}$ where x_1 is set to an arbitrary value (typically $x_1 = (\frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m})$). Go to step b.
b. Selection of a Node. If the set T of nodes to be explored is empty, then stop. Otherwise, choose and remove a node N from T. If $N = (x_1, \tilde{P}, \tilde{Q})$ then go to step c, otherwise $N = (x_2, \tilde{P}, \tilde{Q})$ and go to step d.
c. Direct Feasibility Test (Q). If $\tilde{Q}(x_1)$ is infeasible then go to step b, otherwise choose (x_2, α_1) that optimizes $\tilde{Q}(x_1)$, update (x_1, α_2) that optimizes $\tilde{P}(x_2)$ and then go to step e.
d. Direct Feasibility Test (P). If $\tilde{P}(x_2)$ is infeasible then go to step b, otherwise choose (x_1, α_2) that optimizes $\tilde{P}(x_2)$, update (x_2, α_1) that optimizes $\tilde{Q}(x_1)$ and then go to step e.
e. Branching. Branch on the variable associated to the largest complementarity product $x_{1i}(E_1\alpha_1 - A_{i \cdot} x_2)$ or $(\alpha_2^t E_2 - x_1^t B_{\cdot j}) x_{2j}$:
either $T = T \cup \{(x_2, P^i, \tilde{Q}),\ (x_1, \tilde{P}, Q_i)\}$
or $T = T \cup \{(x_1, \tilde{P}, Q^j),\ (x_2, P_j, \tilde{Q})\}$.
In the case where all complementarity product are already forced to be null, (x_1, x_2) is an equilibrium strategy:
let $T = T \ \cup \ \{(x_2, P^i, \tilde{Q}) : x_{1i} > 0\} \cup \{(x_1, \tilde{P}, Q_i) : A_{i \cdot} x_2 < \alpha_1\}$
$\cup \ \{(x_1, \tilde{P}, Q^j) : x_{2j} > 0\} \cup \{(x_2, P_j, \tilde{Q}) : x_1^t B_{\cdot j} < \alpha_2\}$.
Go to step b.

Steps c and d are the backtracking steps. In each of them, only one problem $\tilde{P}$ or $\tilde{Q}$ is checked for feasibility as only one differs from its father. When this problem is feasible, these steps end with feasible strategies x_1 and x_2 which are vertices of X_1 and X_2. Step e creates new nodes in the tree. When the depth of the current node is less than or equal to $n + m$, two new sons are created. Otherwise,the corresponding solution defines equilibrium strategies since all the complementarity conditions are satisfied, and feasibility is insured by Steps c and d; a new son is created for each strictly positive variable or slack variable.

3.2 $E\chi MIP$ Algorithm

The $E\chi MIP$ algorithm is based on a $0-1$ mixed linear formulation of a bimatrix game, a polymatrix game or the sequential form of a two-persons extensive game. $E\chi MIP$ enumerates all extreme Nash equilibria of these kind of games.

3.2.1 $E\chi MIP$ Pseudo-Code

Step 0: Initialization
Set the following objects as:
- P; $0-1$ mixed linear master program.
- X; Set of coninuous variables in P.
- U; Set of binary variables in P.
- $NE = \emptyset$; Set of Extreme Nash Equilibria.
- $N = 0$; Depth level in the principal branching tree.
- R; Root node of the principal branching tree, with P as current program.
- C; Current node.
- x_i^q, u_i^q; Continuous and binary variables representing the q^{th} strategy $(q = 1, 2, .., m_i)$ of player i $(i = 2, .., n)$.

Set $C = R$ and *Go to Step* 1.
Step 1: Solving and Memorizing
If $N \leq |X|$, Solve the current node's program.
If the prorgam is infeasible *Go to Step* 3.
Else: If a solution $\hat{e}$ is found: If $\hat{e} \notin NE$, add $\hat{e}$ to NE and *Go to step* 2.

Step 2**: Secondary Branching**
If the current node is on the principal branching tree, set the binary variables to $\hat{u}$ ($\hat{u} \in U$) there values in $\hat{e}$ and $\forall\ x_i^q \in X$, such as $\hat{x}_i^q > 0$: Add the branch $x_i^q = 0$ on a new secondary branching current node and *Go to step* 1.
Step 3**: Principal Branching**
If the current node is on the principal tree, then this node will not lead to any equilibrium. **STOP.**
Else, go back to the father node on the principal branching tree and add aconstraint to eliminate the combination $\hat{u}$ of binary variables in $\hat{e}$.
Choose a binary variable $u_i^q \in U$, to branch on, such as no branching was already set on this variable on the preeding nodes and such as x_i^q has the colest value to 0.5 (choose arbitrarily in case of equality).
Set $p = N + 1$:
If $p \leq |X|$ set $N = p$ and:

```
    - Add the branch $u_i^q = 0$,
        If $\hat{u}_i^q = 1$ in $\hat{u}$: delete the elimination constraint on $\hat{u}$,
        Go to step 1.
    - Add the branch $u_i^q = 1$,
        If $\hat{u}_i^q = 0$ in $\hat{u}$: delete the elimination constraint on $\hat{u}$,
        Go to Step 1.
```

Else *Go to Step* 4.
Step 4: END
Display $|NE|$ and all extreme equilibria in NE.

4 XGame-Solver Desciption

At the beginning of our project our main goal was to design a set of algorithms allowing enumeration of all extreme Nash equilibria of some games and to propose some automatic refinement methods for these enumerated equilibria. We tried first of all an enumeration based on the interaction between our algorithms and Cplex optimization software. Importance of numerical precision increased when we tried to have our own implementation of the Simplex algorithm. In fact, implementing exact arithmetics appeared to be problematic without the use of libraries. For our *Xgame-Solver Software* we decided to use an exact arithmetics library that is simple to implement and very efficient. We found a large number of implementations available for the use of the scientific community. While some of them appeared to be difficult or impossible to use with our implementation of the Simplex algorithm, Matt McCutchen's Big Integer Library in C++ proved to be very efficient and fitted perfectly to our project needs.

4.1 Exact Arithmetics Implmentation

In our implementation of exact arithmetic, data is always stored using rationals. A rational is a pair of integers, a numerator and a denominator. The exact arithmetic classes consist of, a BigInteger class, a Rational class, a Simplex class and a Node class.

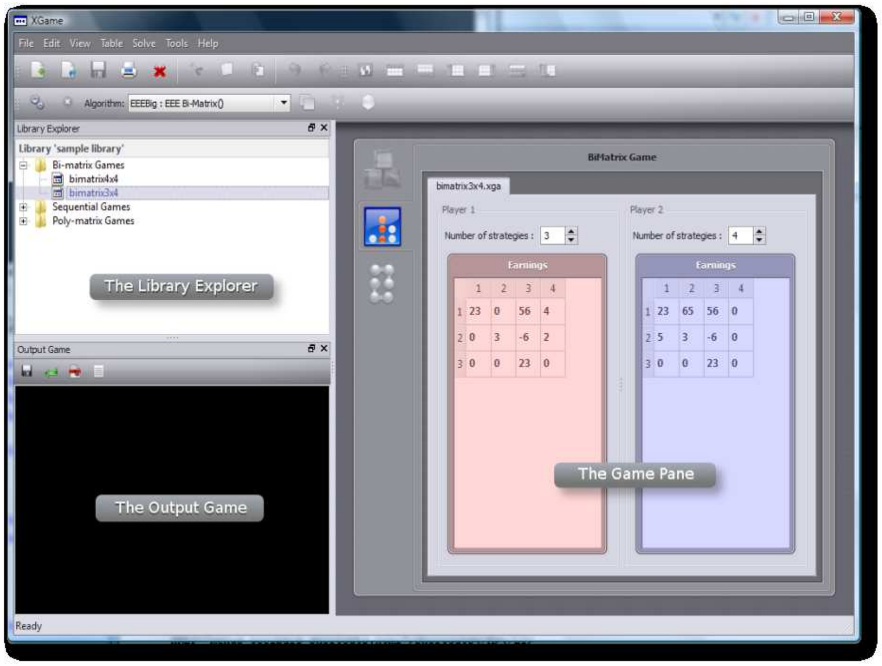
The BigInteger class defines the new type of integers to be used during the enumeration of the extreme equilibria. This class also overloads the elementary operators for these big integers. During the implementation of EEE and EχMIP we observed that it may happen, that the numerator or denominator of a rational exceeds the value of the largest representable integer INTMAX. There is no doubt that the use of this class increases the overall enumeration time. However, the use of exact arithmetics represented a very interesting challenge which made computing time not be considered as the main objective of this work.

The Rational class is based on the Biginteger class. A rational consists of two big integers, a numerator and a denominator. After overloading the elementary operations for these rationals, a Greatest Common denominator function is applied. The Simplex class defines a Dictionary [6] and the set of simplex algorithm (Dantzig, 1951) operations that will be applied to this dictionary in order to find an extreme equilibrium. A dictionary contains an array of rationals. These rationals represent the coefficients of the variables in the dictionary.

The Node class defines a framework for the EEE and EχMIP algorithms. Each node contains the current dictionary and a pointer to its father. This class contains branching methods that permits to obtain a certain number of sons nodes from a father node. In general, EEE can be used to enumerate alle xtreme equilibria of a bimatrix game or a two person extensive game. The EχMIP algorithm can be used in order to enumerate all extreme equilibria of a bimatrix game, a three person polymatrix game, or the sequence form of a two person extensive game. The XGame Solver software uses this implementation of the EχMIP algorithm and is available on its site http://www.XGame-Solver.net for free download.

4.2 Main User Interface

To enable the scientific community benefit from our work we tried to develop a user oreinted application that is very easy to understand and to use. Figure 1 presents the Main interface of the XGame-Solver software.



Algorithm: EEEBig : EEE Bi-Matrix()

Player 1 — Number of strategies : 3

	1	2	3	4
1	23	0	56	4
2	0	3	-6	2
3	0	0	23	0

Player 2 — Number of strategies : 4

	1	2	3	4
1	23	65	56	0
2	5	3	-6	0
3	0	0	23	0

Figure 1: XGame main interface

First of all, one can notice that users can choose to load any existing game file with ".xga" (XGame) extension or create a new game file. Once a game file is open, users may choose to edit the content of the payoff matrices before solving.

4.3 Create New Game

Users may decide to create their own game files using the *Create new game* icon. This icon calls an interface to be used to define the kind of game to create and the different size parameters.

4.4 Create and Explore Library

Users may create their own game library or explore an existing one using the *File* menu and then the *Library* menu.

4.5 Edit Solvers

User may also view and edit the set of algorithms assigned to each type of games. This interface contains also a brief definition of each kind of game to help beginners find the best answers to their needs.

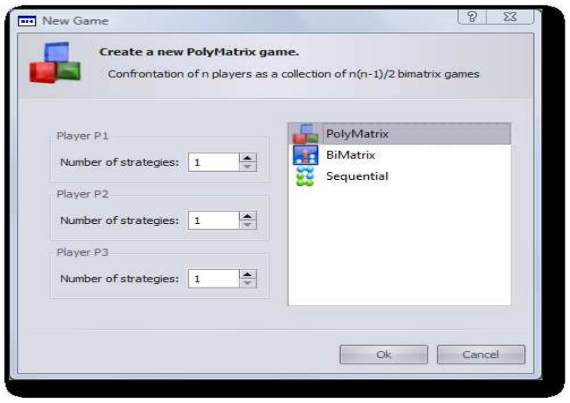


Figure 2: Create new game

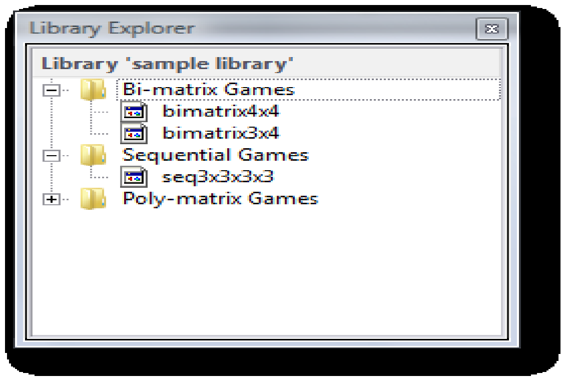


Figure 3: Explore library

Our intention is also to extend the range of abilities of *XGame Solver* by implementing other algorithms solving Bimatrix games and other kinds of games. We believe that it would be very helpfull for the scientific community.

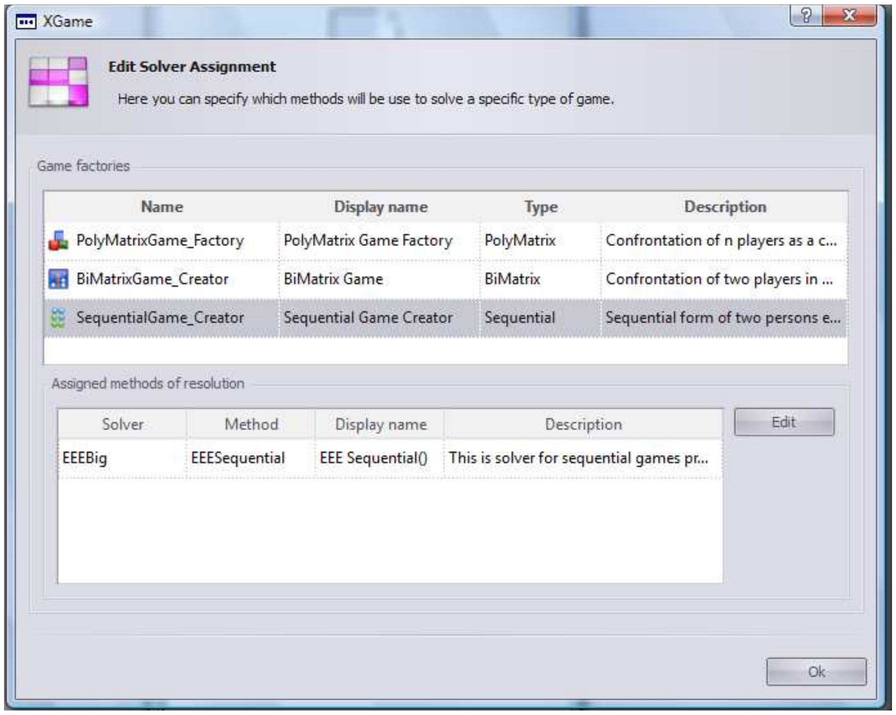
Concerning bimatrix games and sequential forms of two person extensive games, users may choose to enumerate all extreme Nash equilibria using *EEE* or $E\chi Mip$. One can notice also that users may solve bimatrix games, sequential forms of two person extensive games and polymatrix games simultaneously. Several runs of the algorithms could be launched on different examples at the same time. Users have also the possibility to stop any running process during its execution using the *Red X* icon.

4.6 Edit Icons

In addition to the classical edition icons: *Copy, Cut, Paste*, we added some other icons to the edit menu: *Add row above, Add row below, Add column left, Add column right, Remove rows, Remove columns.* There is also a possibility to apply a predefined mathematical tranformation on a matrix using: *Transpose, Identity, Factor.*

4.7 Output window

Once an enumeration process is executed the output of the chosen algorithm appears in the bottom window. Users have the possibility to hide or show this window. They can also enlarge this window and scroll up and down or change its font. Finally, users have the possibility to save the output giving a name of their choice to the ouput file.



Here you can specify which methods will be use to solve a specific type of game.

Game factories

Name	Display name	Type	Description
PolyMatrixGame_Factory	PolyMatrix Game Factory	PolyMatrix	Confrontation of n players as a c...
BiMatrixGame_Creator	BiMatrix Game	BiMatrix	Confrontation of two players in ...
SequentialGame_Creator	Sequential Game Creator	Sequential	Sequential form of two persons e...

Assigned methods of resolution

Solver	Method	Display name	Description
EEEBig	EEESequential	EEE Sequential()	This is solver for sequential games pr...

Figure 4: Edit solvers

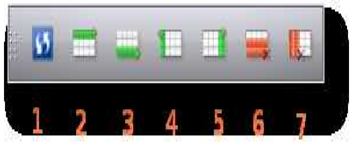
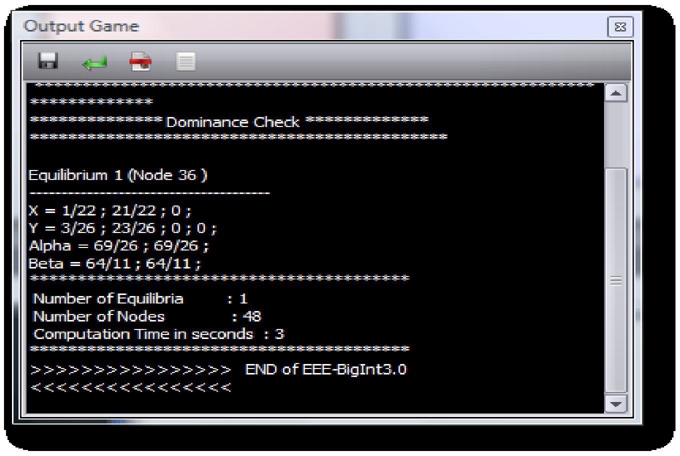


Figure 5: Tool bar



```
*************************************************************
**************
*************** Dominance Check **************
**********************************************

Equilibrium 1 (Node 36 )
----------------------------------------
X = 1/22 ; 21/22 ; 0 ;
Y = 3/26 ; 23/26 ; 0 ; 0 ;
Alpha = 69/26 ; 69/26 ;
Beta = 64/11 ; 64/11 ;
*****************************************
 Number of Equilibria       : 1
 Number of Nodes            : 48
 Computation Time in seconds  : 3
*****************************************
>>>>>>>>>>>>>>>>>  END of EEE-BigInt3.0
<<<<<<<<<<<<<<<<<
```

Figure 6: Output window

4.8 Post Processing

XGame provides three post-processings by default: Nash Subsets, Quasi-Strong and Perfect. Nash Subset post-processing identifies maximal Nash subsets of bimatrix games. Quasi-Strong post-processing detects extreme quasi-strong equilibria and Perfect post-processing detects extreme perfect Nash equilibria.

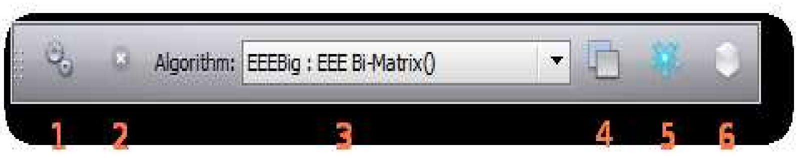


Figure 7: Solver bar

5 Conclusion

The *XGame Solver* software contains EEE and $E\chi MIP$ algorithms implementations with exact arithmetics. We look forward to make the scientific community benefit from this package. Future extensions of our application may include implementing other equilibrium computation algorithms and other kinds of games. Users may also implement their own solvers and define their own games to be incorporated using plugins to the XGame Solver Software.

References

[1] Audet C., Hansen P., Jaumard B., Savard G.: (2001), "Enumeration of all extreme equilibrium strategies of bimatrix games", *SIAM Journal on Scientific Computing*, Vol. 23, No. 1, 323–338.

[2] Audet C., Belhaiza S. Hansen P.: (2006), "Enumeration of all Extreme Equilibria in Game Theory: Bimatrix and Polymatrix Games", *Journal Of Optimization Theory and Applications*, Vol. 129, No. 3, June.

[3] Audet C., Belhaiza S. Hansen P.: (2007), "Perfect and Proper refinements of all extreme Nash equilibria for Bimatrix games", *Les Cahiers du GERAD*, G–2007–85, November.

[4] Audet C., Belhaiza S. Hansen P.: (2009), "A new sequence form approach for the enumeration and refinement of all extreme Nash equilibria for extensive form games", *International Game Theory Review*, Vol. 11, No. 4.

[5] Audet C., Belhaiza S. Hansen P.: (2010), "A note on Bimatrix Game Maximal Selten Subsets", *Les Cahiers du GERAD*, G–2010–03, January.

[6] Chvátal,V.: (1998), "Linear programming," *New York*, Freeman.

[7] Júdice J.J., Mitra G.: (1988), "Reformulations of mathematical programming problems as linear complementarity problems and investigation of their solution methods", *Journal of Optimization Theory and Applications*, Vol. 57, 123–149.

[8] Koller D., Megiddo N.: (1992), "The complexity of two-person zero-sum games in extensive form", *Games and Economic Behavior*, 4, 528–552.

[9] Kuhn H.W.: (1953), "Extensive games and the problem of information", in: *Contributions to the theory of Games II*, H.W. Kuhn and A. Tucker (eds.), Annals of Mathematics Studies, 28, Princeton Univ. Press, 193–216.

[10] Millham C.B.: (1974), "On Nash Subsets of Bimatrix Games", *Naval Research Logistics Quarterly*, Vol. 21, No. 2, 307–317.

[11] Nash J.F.: (1950), "Equilibrium Points in n-Person Games", *Proceedings of the National Academy of Sciences*, Vol. 36, No. 1, 48–49.

[12] Romanovskii I.V.: (1962), "Reduction of a game with complete memory to a matrix game", *Soviet Mathematics*, 3, 678–681.

[13] von Stengel B.: (1996), "Efficient computation of behavior strategies", *Games and Economic Behavior*, 14, 220–246.

[14] von Stengel B., van den Elzen A., Talman D.: (2002), "Computing normal form perfect equilibria for extensive two-person games", *Econometrica*, Vol. 70, No. 2, 693–715.