Cent. Eur. J. Comp. Sci. • 2(2) • 2012 • 128-142
DOI: 10.2478/s13537-012-0008-6

VERSITA

Central European Journal of **Computer Science**

A new evolutionary approach for computing Nash equilibria in bimatrix games with known support

Research Article

Urszula Boryczka[1*], Przemyslaw Juszczuk[1†]

1 *Institute of Computer Science, Silesian University*
ul.Bedzinska 39, 41-200 Sosnowiec, Poland

Received 20 December 2011; accepted 02 March 2012

Abstract: In this paper, we present the application of the Differential Evolution (DE) algorithm to the problem of finding approximate Nash equilibria in matrix, non-zero sum games for two players with finite number of strategies. Nash equilibrium is one of the main concepts in game theory. It may be classified as continuous problem, where two probability distributions over the set of strategies of both players should be found. Every deviation from the global optimum is interpreted as Nash approximation and called ϵ-Nash equilibrium. The main advantage of the proposed algorithm is self-adaptive mutation operator, which direct the search process. The approach used in this article is based on the probability of chosing single pure strategy. In optimal mixed strategy, every strategy has some probability of being chosen. Our goal is to determine this probability and maximize payoff for a single player.

Keywords: ϵ-Nash equilibrium • strategy support • differential evolution

1. Introduction

Game theory, the study of strategic interactions among rational agents, has had a great impact on many sciences such as economics [1], evolutionary biology [10], political science [23], and computer science. Close connection beetween game theory and decision theory allows the use of concepts related to game theory in decision making. The Nash equilibrium is one of the most important concepts in game theory, forming the basis of much recent work in the multiagent decision making, artificial inteligence [32] and electronic marketplaces [11]. As such, effciently computing Nash equilibria is one of the most important problems in the computational game theory. In this article we propose a new approach for the evolutionary generating approximate Nash equlibria. Our algorithm is based on the well known John Nash theorem which says that every game has at least one Nash equilibrium in mixed strategies [21]. We consider two person, non-cooperative games in the strategic form with random and normalized payoffs where every player has finite number of strategies to choose. Since it was shown that finding a Nash equilibrium is PPAD-complete [5], even for 2-player

* *E-mail: urszula.boryczka@us.edu.pl*
† *E-mail: przemyslaw.juszczuk@us.edu.pl*

games [4], the question of approximate Nash equilibrium emerged as the central remaining open problem in the area of equilibrium computation. We assume that all payoffs have been normalized to values between 0 and 1. This is a common assumption, since scaling the utilities of a player by any positive factor, and applying any additive constant, results in an equivalent game.
In general, game Γ in the strategic form for two players is formally described as a couple:

$$\Gamma(R, C) \tag{1}$$

and consists of two payoff matrices $C \in R^{M \times N}$ for the row player and $R \in R^{M \times N}$ for the column player. M and N denote, respectively, the number of the strategies for the first and the second player. We consider bimatrix games, so we also have a set of players with two elements $\{X, Y\}$, where X is called the row player, and Y is called the column player. Every player has a finite set of strategies: $X = (x_1, x_2, ..., x_m)$ for the first player, and $Y = (y_1, y_2, ...y_n)$ for the second player. Single strategies of both players are often called pure strategies, in constrast to mixed strategies. We define mixed strategy for the X player as the m-dimensional vector:

$$x = (P(x_1), P(x_2), ..., P(x_m)),$$

where:

$P(x_i)$ - a probability of choosing strategy x_i.

In other words, a mixed strategy for a player is a vector of probabilities for playing single strategies. We also introduce the concept of the support. The support of a mixed strategies is the set of pure strategies which have positive probability. Both mixed strategies for the column, and the row player are called strategy profile. Support of the mixed strategy x is a subset M_x:

$$\forall_i, x_i \in M_x, P(x_i) > 0$$

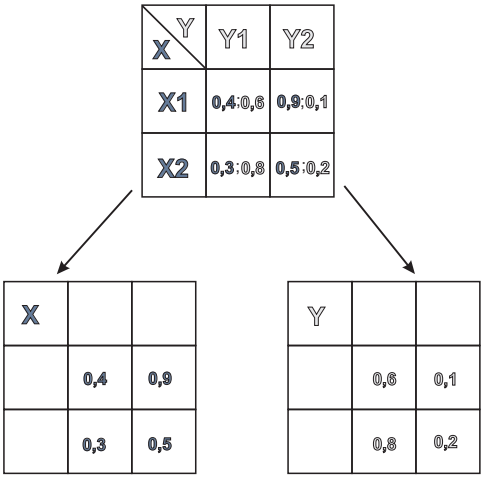

Figure 1. An example of the 2 player game in the strategic form.

Every bimatrix game (Fig. 1) consists of two joined payoff matrices: numbers on the left (in each cell) reflect the X player payoff and numbers of the right - the Y player payoff. Players choose their moves simultaneously (X player chooses one of the rows, and Y player chooses one of the columns). Values at the intersection - cell of the bimatrix correspond to payoffs for players.
Our motivation is to show a new evolutionary method of finding Nash equilibrium in non-zero sum games with known support. We proposed the Differential Evolution algorithm (DE) [30]. The Differential Evolution has a very strong mutation schema, which directs search process to the global optimum. Mutation is the major genetic operator. It is not a trivial process and it also provides the algorithm convergence. Moreover, the mutation is performed before the crossover process. The proposed approach will be explained in details in Section 4.

Our article is organized as follows: first, we give some examples of other algorithms (including mathematical methods) for computing exact and aproximate Nash equlibria. In the next section we give a brief description of the problem. In the Section 4 we describe the DE algorithm and explain details of the solution. Section 5 contains the experiments and results. We summarize with short conclusions.

2. Related works

The main algorithm for computing Nash equilibria is the Lemke–Howson (LH) algorithm [16], which is a pivoting algorithm similar to the simplex algorithm for linear programming. The algorithm is started by choosing a pure strategy of one of the players. For each such choice the algorithm progresses deterministically, giving rise to a different LH path, which terminates at some equilibrium of the game. Unfortunatelly, the classical algorithm by Lemke and Howson (1964) not always returns every solution in the game. The second, modified implementation of the algorithm was proposed in 1991 [13]. The algorithm as described by one of the authors is based on the original version invented by Lemke and Howson. However, it differs from this version with respect to several features. It works directly with the matrices defining the bimatrix game. It has an easy and very direct geometrical interpretation. In the same year, other, very computational expensive approach was proposed [12]. In 2004 a new, faster algorithm using the support enumeration was proposed [26]. Mentioned algorithm is similar to the previous one, but for example it checks dominant strategies. It is very fast, especially for the games with small support. That approach was also considered as parallel [34]. Authors considered the problem of computing all Nash equilibria in bimatrix games. The algorithm computes all Nash equilibria by searching all possible supports of mixed strategies. One year later in 2005 an approach based on the mixed integer programming was described [28]. Other methods involve using graphs [33]. In particular, the general two-person problem is reduced to an indefinite quadratic programming problem of special structure involving the $n \times n$ adjacency matrix of an induced simple graph specified by the input data of the game, where n is the number of players strategies. There were also several attempts to develop algorithms oriented at finding pure Nash equilibria [31], but this problem seems to be less significiant (pure Nash equilibria are far easier to find than mixed Nash equilibria). Other type of the equilibrium: well-supported Nash equilibrium has been presented in [24]. As for polynomial time algorithms, it is fairly simple to obtain a $\frac{3}{4}$-approximation and even better a $\frac{1}{2}$-approximation [6]. An improved approximation of the $\epsilon-$Nash equilibrium for $\epsilon \approx 0.38197$ was obtained by Daskalakis, Mehta and Papadimitriou [7]. In [29] P. Spirakis and H. Tsaknakis have obtained another algorithm achieving an improved approximation of 0.3393. The concept of the adaptive algorithm is used in the GAMBIT program [19] and it is based on the quantum response equilibrium (QRE). Above articles focus on finding Nash equilibria, when we do not know anything about the game. In [20] the method for calculating expected payoffs for both players was proposed. This property may be used to find the Nash equilibrium. As far as we know, it's the first approach for computing Nash equilibria, when support for both players is given. This problem may be considered as somewhat simpler than finding the Nash equilibrium without given support. The roposed approach may be considred as unique, because it is based on the equation defined by I. Milchtaich in his article. We present it with details in next sections.

3. Problem formulation and definitions

A non-cooperative game in strategic form consists of a set of players, and, for each player, a set of strategies available to him as well as a payoff function mapping each strategy profile (i.e. each combination of strategies, one for each player) to a real number that captures the preferences of the player over the possible outcomes of the game. Recalling the support definition: support of the mixed strategy x is a subset M_x:

$$\forall_i, x_i \in M_x, P(x_i) > 0$$

where:

$P(x_i)$ - the probability of choosing strategy x_i. Now we can define the Nash equilibrium concept. Nash equilibrium is a strategy profile x^*, y^* such that no deviating player could achieve a payoff higher than the one that the specific profile gives him. In other words, Nash equilibrium definition is proper for every pure strategy in the support. Mixed strategy

profile for two players may be defined by the following equations:

$$\forall_i, x_i \in M_x, x_i R y^* \leq x^{*T} R y^* \tag{2}$$

$$\forall_i, y_i \in N_y, x^{*T} C y_i \leq x^{*T} C y^*, \tag{3}$$

where:

x_i - the i-th pure strategy of the X player;
$x_i R y^*$ - the payoff for the Y using his mixed strategy y^* against the i-th pure strategy of the player X;
M_x,N_y - support for the X and Y players;
x^{*T} - transposed vector specifying the mixed strategy of the X;
R - the Row player;
C - the Column player.

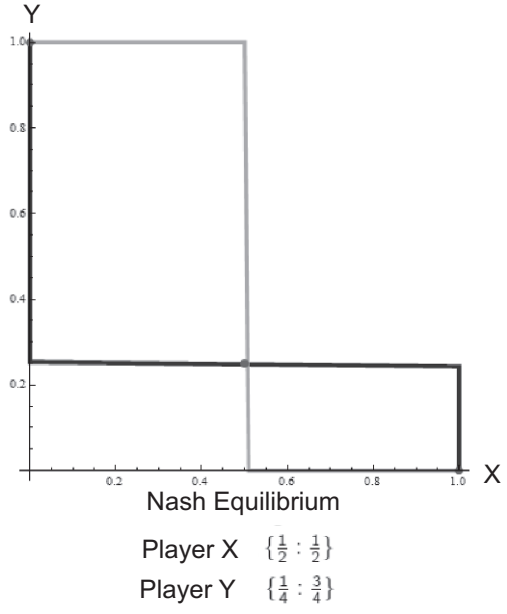

Figure 2. Simple Nash Equilibrium Example.

In Fig. 2 we can see visualisation of the simple Nash equilibrium. The X axis represents the probability of choosing the first strategy of player X. The probability of the second strategy may be computed as follows: $P(x_2) = 1 - P(x_1)$. The same assumption applies to the second player. Despite the certain existence of such equilibria [21], the problem of finding any Nash equilibrium even for games involving only two players has been recently proved to be complete in the PPAD class. PPAD is a class of total search problems, that is, problems for which solutions are guaranteed to exist, and the challenge is to actually exhibit a specific solution. This fact emerged the computation of approximate Nash equilibria, referred to as ϵ-Nash equilibria. For any $\epsilon > 0$, we define ϵ-Nash equilibrium as:

$$\forall_i, s_i \in M_x, s_i R y^* \leq x^{*T} R y^* + \epsilon, \tag{4}$$

$$\forall_i, t_i \in N_y, x^{*T} C t_i \leq x^{*T} C y^* + \epsilon. \tag{5}$$

An ϵ-Nash equilibrium is a strategy profile such that no deviating player could achieve a payoff higher than the one that the specific strategy profile gives him, plus ϵ.

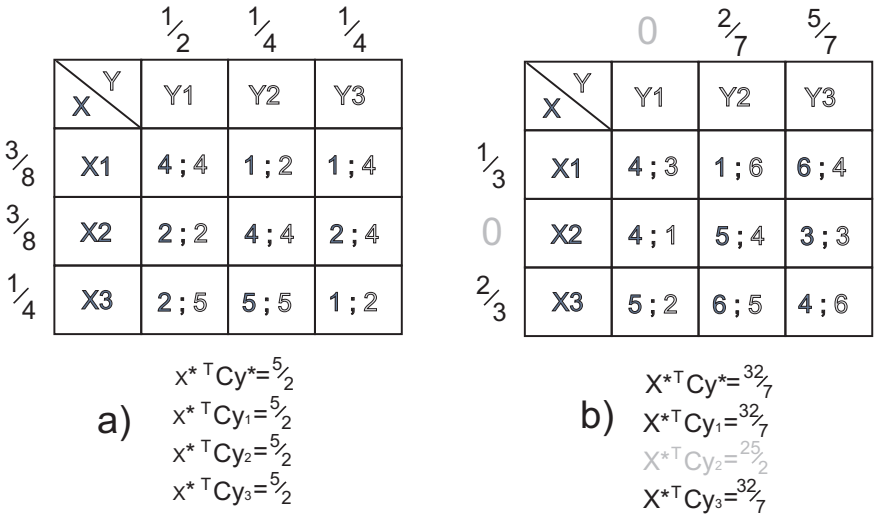

Figure 3. Two examples of games. a - with all strategies active. b - some strategies with probability of chosing equal to 0.

In games where players use every pure strategy (Fig. 3 a), so set X is equal to M_x and set Y is equal to N_y the following equality is fulfiled:

$$\forall_i, x_i R y^* = x^{*T} R y^*, \tag{6}$$

$$\forall_i, x^{*T} C y_i = x^{*T} C y^*. \tag{7}$$

As we can see, when using optimal mixed strategy against any pure strategy of second player the payoff is always the same. Above equalities are not satisfied when set $X \neq M_x$ or set $Y \neq N_y$ (Fig. 3b). Every game may be transformed into the game with known support for both players, when this support is given (Fig. 4).

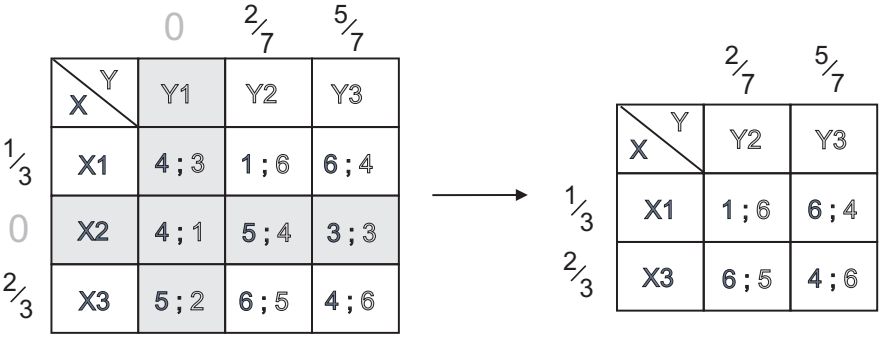

Figure 4. Strategies with 0 probability of chosing are skipped/

Our problem may be described as finding probability distribution over the set of pure strategies of both players satisfying Eqs. 2 and 3.

The above idea is based on the I. Milchtaich article [20]. The author proposed a method for computing payoffs in bimatrix games with completely mixed Nash equilibria. It should be noted that it is possible to compute Nash equilibrium, if we know exact payoff for both players. In any bimatrix game in which the payoff matrix of one of the players is given by A, and in any equilibrium in the game in which this player's strategy is completely mixed, the player's payoff λ is given by the formula:

$$\frac{\lambda - x}{\lambda - y} = \frac{|A - x \cdot E|}{|A - y \cdot E|} \tag{8}$$

where A is a square matrix, and x and y a pair of distinct real numbers with $|A - y \cdot E| \neq 0$.
Unfortunately I. Milchtaich did not present any deep study of this problem. We treat the above equation as a starting point. On the basis of the assumptions from this section we were able to derive a method for computing completely mixed Nash equilibria. In the next section we present details of the proposed solution. It shouldn't be treated like a stand-alone method. Our main intention is rather to present a method which will be the basis of the algorithm to computing mixed Nash equilibria. We want to provide a method based on the Diffferential Evolution, which could be successfully compared with state-of-the-art algorithms like the Lemke Howson algorithm.

4. Proposed solution

In the last decades many deterministic and stochastic methods for optimization problems have been developed. However, no universal technique which could give good results for all optimization problems has been found yet. Different approaches are used to find the global optimum for multimodal function. Since deterministic methods are often too weak or too slow, stochastic methods are used. These methods can find only approximate solutions, but in many applications this approach is sufficient. Algorithms of this type often take inspiration from biological or social behavior. The ase of these techniques is a population of individuals which is improved in subsequent iterations. The Differential Evolution algorithm was chosen from other evolutionary approaches. This method is dedicated to the continuous function optimization problem. We shown that problem of finding mixed Nash equilibrium may be treaten as the continuous problem. In this context, the Differential Evolution algorithm seems to be the most proper way to solve the above problem.
Differential evolution (DE) is a stochastic, population-based search strategy developed by Storn and Price in 1995, and deeply studied by Jouni Lampinen, Ivan Zelinka [14, 15, 27], and others. Mainly it has been applied to optimize functions defined over continuous-valued landscapes. DE has also been applied to train neural networks [18]. In this case an individual represents a complete NN. Similar approach was made for training Fuzzy Cognitive Maps [9]. Other applications of Differential Evolution focus on clustering [25] or image analysis [17].
As in any other evolutionary algorithms, before the population can be initialized, both upper and lower bounds for each gene of the genotype must be specified. After that, the selection process takes place. During the selection stage, three parents are chosen and they generate a single offspring which competes with a parent to determine who passes to the following generation. DE differs from these evolutionary algorithms in that:

- mutation is a primary operator, and crossover is an auxiliary operator,
- mutation is applied first to generate a trial vector, next this vector is used in crossover procedure,
- mutation step sizes are not sampled from a prior known probability distribution function,
- selection always favor better individual.

The DE algorithm begins with the initialization of population $P(0)$ which consist of n_X individuals. The initialization consists of a random distribution of individuals. The population should be distributed uniformly, which provides a good sampling of search space. Over time, as the search progresses, the distances between individuals become smaller, with all individuals converging to the same solution. In the main loop of the algorithm some actions which should improve the population are performed. For the each individual (vector), firstly, its fitness is evaluated. Then the mutation process follows. The pseudocode of the general DE algorithm is presented below:

1. Create the initial population of genotypes $S = \{\vec{s}_1, \vec{s}_2, ..., \vec{s}_n\}$,
2. Set the generation number $g = 0$
3. Until the stop criterion is not met:
   (a) Compute the fitness function for every genotype in the population $\{f(\vec{s}_1), f(\vec{s}_2), ..., f(\vec{s}_n)\}$
   (b) Create the population of trial genotypes V_g based on S_g

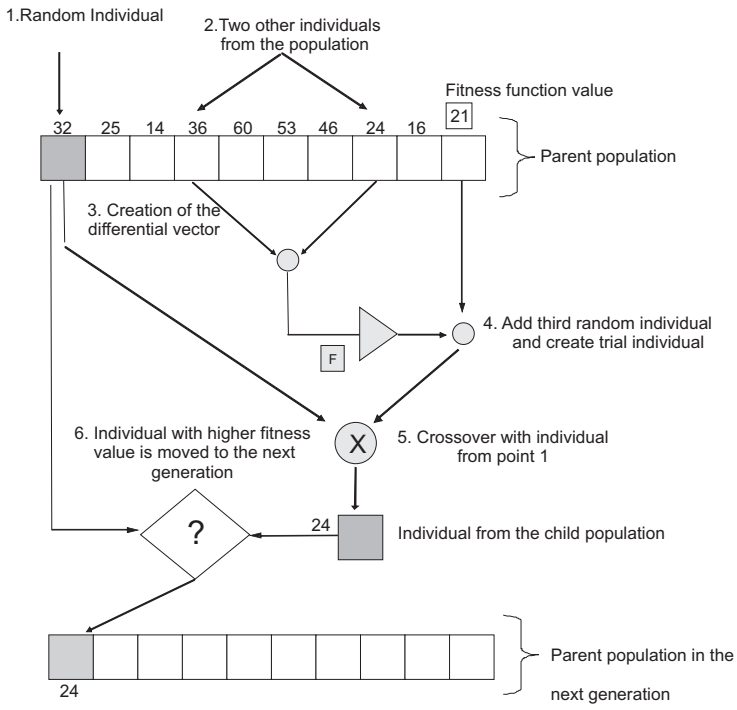

Figure 5. DE Schema.

(c) Make crossover of genotypes from the population S_g and V_g to create population U_g

(d) Choose the genotypes with the highest fitness function from population U_g and S_g for the next population

The problem of computing Nash equilibria may be clasified as the continuous function optimization problem. In game theory a player is said to use a mixed strategy whenever he or she chooses to randomize over the set of available actions. Formally, a mixed strategy is a probability distribution that assigns to each available action a likelihood of being selected. Every action is a row or a column of the given payoff matrix, so it's simple way to create the population of individuals on the basis of the payoff matrix (Fig. 6). For example, 2 player game, where each of the players has 20 pure strategies is 40-dimensional optimization problem. Every gene in genotype has a value in the range $\langle 0, 1\rangle$ and represents the probability of choosing the pure strategy.

The DE mutation operator produces a trial vector for each individual of the current population by mutating a target vector with a weighted differential. We shown that in the strategic form game probabilities of choosing single strategy of the player may be represented as the vector. This vector (also called individual) is treated like a standard individual in the basic DE. In DE, mutation step sizes are influenced by differences between individuals of the current population:

$$\forall_i \ \forall_j \ v_{i,j} = s_{r_1,j} + F \cdot (s_{r_2,j} - s_{r_3,j})$$

where $s_{r_1,j}$ is the j-th element of the individual r_1 in the parent population S. The parameter F specifies the strength of impact of the difference vector (between the two genotypes from the population). r_1, r_2, and r_3 are three randomly selected indexes of individuals. Each increment moves selected individuals towards the global optimum. Note that the mutation is successively subjected to each genotype within the population. An individual v_i represents individual after mutation. $(s_{r_2} - s_{r_3})$ is a differential vector created from the two random individuals s_{r_2} and s_{r_3}.

Like for many other evolutionary algorithms, also for Differential Evolution many modifications were developed. The most often modified elements of DE algorithm are:

- a method of target vector selection (denoted as x),
- a number of differential vectors used for trial vector creation (y),

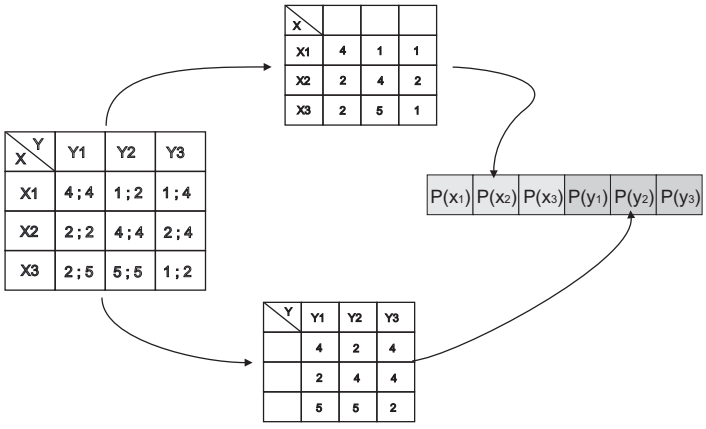

Figure 6. Creation of the single individual.

- a crossover method (z).

Each of the DE algorithms can be described by strategy DE/x/y/z.

- Strategy I: DE/rand/1/z It is the DE strategy which characterises the basic DE algorithm. It uses random (rand) selection of target vector and only one differential vector for the trial vector creation. Like other strategies also this one can use binomial (bin) or exponential (exp) crossover. In the DE/rand/1/z strategy the trial vector is calculated form the equation:

$$u_i(t) = x_{i_1}(t) + F \cdot (x_{i_2}(t) - x_{i_3}(t))$$

- Strategy II: DE/best/1/z

  In this strategy the best individual $\hat{x}(t)$ from the current population is the target vector. In this case the trial vector $u_i(t)$ is calculated as follows:

$$u_i(t) = \hat{x}(t) + F \cdot (x_{i_2}(t) - x_{i_3}(t))$$

- Strategy III: DE/x/n_v/z

  n_v signifies the number of differential vectors used for trial vector $u_i(t)$ creation. Trial vector in this strategy is calculated from the following equation:

$$u_i(t) = x_{i_1}(t) + F \cdot \sum_{k=1}^{n_s} (x_{i_2,k}(t) - x_{i_3,k}(t))$$

  $(x_{i_2,k}(t) - x_{i_3,k}(t))$ denotes the k^{th} differential vector ($k \in \{1, 2, \ldots, n_v\}$). The larger the value of n_v, the better the search space exploration. Unfortunately, the more differential vectors are used in the mutation, the greater the complexity of the algorithm.

Above we presented three most popular differential evolution schemas. Nevertheless, in our research we used the basic mutation. We want to show, that high effectiveness of the algorithm does not depend on the selected schema.
The crossover process consists of the creation of a new individual (offspring) u_i. Some of the elements of the vector u_i are derived from individual p_i and the others from the trial vector v_i. The crossover operation operates both the genotype from the population S and the trial genotype (population V). In the process of crossing over, the parameter $CR\langle 0, 1\rangle$ and the randomly-chosen number are used.

$$\forall_i \forall_j u_{i,j} = \begin{cases} v_{i,j} & \text{when } RAND[0,1) < CR, \\ p_{i,j} & \text{in other case.} \end{cases}$$

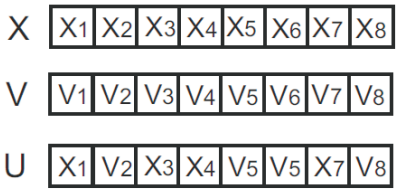

Figure 7. Crossover schema.

Simple crossover schema may be seen in the Fig. 7.
For the simulation of the evolution, we apply fitness function given below:

$$fitness = \underbrace{\sum_{i=1}^{n} |x^{*T} R y * - x^{*T} R y_i|}_{\text{Payoff deviation } X} + \underbrace{\sum_{i=1}^{n} |x^{*T} C y * - x_i R y^*|}_{\text{Payoff deviation } Y} + c_1 \cdot \underbrace{|\sum_{i=1}^{n} P(x_i) - 1|}_{\text{probability factor } X} + c_2 \cdot \underbrace{|\sum_{i=1}^{m} P(y_i) - 1|}_{\text{probability factor } Y} \quad (9)$$

where:
$x_i R y^*$ - the payoff for the Y using his mixed strategy y^* against the i-th pure strategy of the player X;
c_1 - scalability factor for the X player equal to $dim X \cdot 10$, $c_1 = 60$;
c_2 - scalability factor for the Y player equal to $dim Y \cdot 10$, $c_2 = 60$;
$P(x_i)$ - probability of choosing pure $i - th$ strategy of the X player;
$P(x_i)$ - probability of choose i-th pure strategy of X player.

In the above Eq. 9 we can separate the fitness function into four parts. Payoff deviations (payoff deviation X and payoff deviation Y) are calculated on the basis of the expected payoff. If each player uses his best mixed strategy, both factors are equal zero. The sum of all probabilities for each player should be equal to one, so two other parts of the fitness function (probability factor X and probability factor Y) should be equal to zero. The fitness function is computed for each individual from population U. The genotype with the lower fitness function value is transferred to the next population. Fitness function equal to 0 is identified as global optimum - Nash equilibrium. To construct the population for the next generation, deterministic selection is used: the offspring replaces the parent if the fitness of the offspring is better than its parent; otherwise the parent survives to the next generation. This ensures that the average fitness of the population does not deteriorate. The best individual derived from the last generation is the result of the DE algorithm.

5. Parameter tuning in the differential evolution

The Differential Evolution is the algorithm which requires to set a few parameters. In this section we describe in details these parameters and their impact on the algorithm convergence. The most important parameters are:

- F mutation factor - parameter specifying the impact of difference between two individuals at the value of the newly created individual genotype (trial genotype). Large value increases the randomness of the search. In the literature, F is often equal to 0.7. However, it's important to check, if this value is proper in the problem, where every gene in the genotype it's in the range of $\langle 0, 1 \rangle$.
- CR crossover factor - The crossover process uses both the individual from the initial population, as well as the trial individual. CR determines, which parts from the trial genotype are inherited by the child individual.
- K number of the individuals - there where a few examples of determine the value K:
$10 \cdot dimension - problem$ or even $50 \cdot dimension - problem$. In case, where every fragment of the genotype is within value $\langle 0, 1 \rangle$, this value should be decreased.

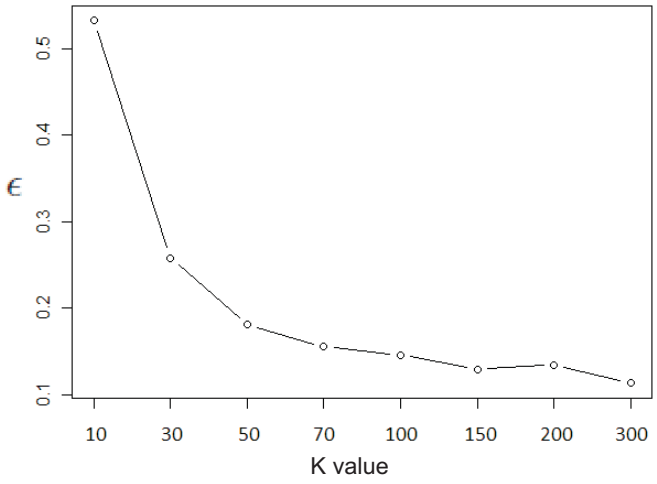

Figure 8. Relation between population size K and ϵ value.

There where also some minor settings that may lead to the fast convergence. For example:

- λ modification - greatly increases convergence of the population. This method was described in [2]. At this moment, accurate value of the λ parameter in unknown, so it was excluded from the parameters.
- mutation schema - Differential Evolution is the algorithm, which has several different mutation schemas that work better on selected problems.

We decided not to invole any additional settings, which may significantly affect the results of the experiments. Below experiments were computed for the arbitrarily selected problem, and every experiment was repeated 50 times. All tests were performed for the games where the number of strategies for the every player is equal to 3 - it gives us 6-dimensional function.
We considered population size in the range $\langle 10, 300 \rangle$. As it may be seen in Fig. 8, the lowest ϵ value was obtained for the K equal to 300. Higher number of individuals significantly improves algorithm convergence. On the other hand, large populations lengthen overall computation time. We used the same experiment, to define the largest acceptable computation time and results were presented on the Fig. 9.
Desirable computation time, and the ϵ values were set to:

- $\epsilon < 0.15$;
- $time < 10s$.

Above values were selected arbitrarily. In both cases, those values have been achieved for the K size equal to 100. This value is much lower than suggested in other articles. Interestingly, up to 300 individuals, time - population dependency increases almost linear. This assumption wasn't tested for the larger population size, but above fact seems te be helpful especially for the significantly larger games, where number of strategies is larger than 3.
The second important parameter is the CR. We tested different values in the range $\langle 0.1, 0.9 \rangle$ with step equal to 0.1. Results may be seen in Fig. 10. It is clear to see that small CR values give far more better results. Small CR values affect on the convergence speed. Much of the trial individual genotype is rejected in favor of the parent individual.
The last tuned parameter was mutation factor F. We tested different values in the range $\langle 0.05, 1.0 \rangle$ with step equal to 0.05. As it may be seen in Fig. 11, the best results were obtained for the $F > 0.5$. F parameter should not be set to the values greater than 0.8 - it significantly reduces the convergence in the final phase of the algorithm (the last few hundreds iterations).
Parameter tunning for the differential evolution algorithm was an important part of the experiments. According to our knowledge, this article is the first approach to using differential evolution algorithm for computing the Nash Equilibria

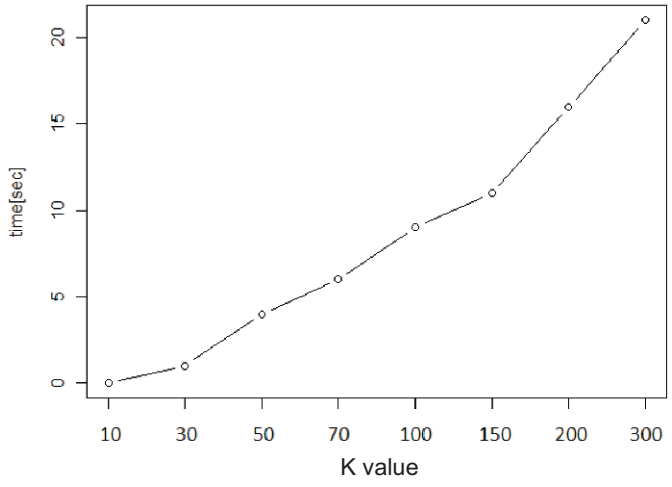

Figure 9. Parameter tuning for the K - time dependency

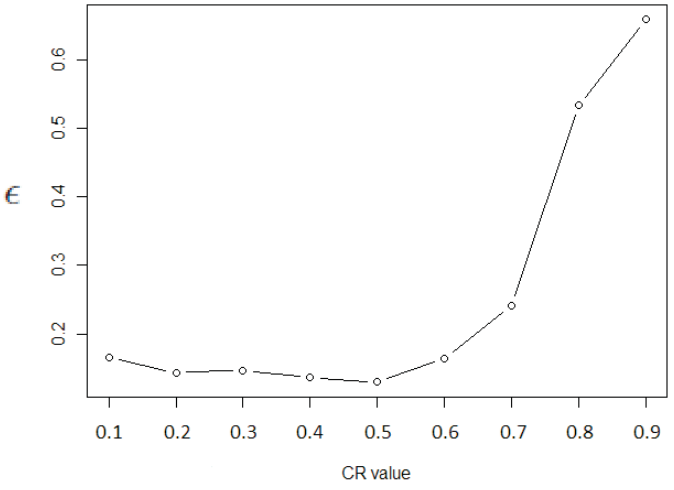

Figure 10. Parameter tuning for the CR factor.

with known support. Parameter tunning for the above algorithm was the subject of many articles [27]. Our goal was to confirm or reject proposed in the literature values. Values of the parameters F and CR, as expected, agree with the values proposed in the literature. An important conclusion is that the value of the parameter K should be significantly reduced. The problem of computing the Nash equilibrium is very complex, but the search space is limited. It allows to significantly reduce the number of individuals in the initial population S.

6. Experiments

The aim of this research work is to determine if the differential evolution algorithm is capable to find and rate the set of optimal mixed strategies. The DE algorithm has the following parameters:

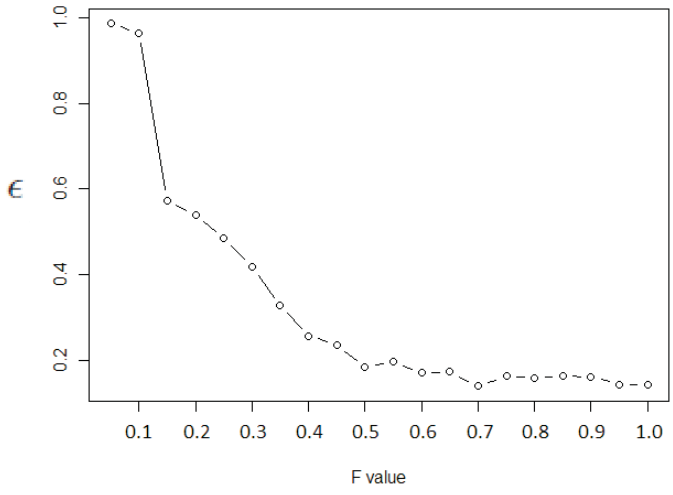

Figure 11. Parameter tuning for the F.

- binomial crossover;
- crossover parameter $CR = 0.5$;
- mutation parameter $F = 0.7$;
- the population size $K = 100$;
- the number of differential vectors $n_v = 1$.

For our experiments we used a 10 different random games with normalized payoffs generated with GAMUT [22]. It is a standard test suite used in game theory and it allows to generate different classes of games. The number of strategies for every player is equal to 3 - it gives us 6-dimensional function. In the Table 1 we present ϵ values for 10 different games. For every game, the algorithm was started 50 times, and results are averaged. As we can see, minimal error for every game was smaller than 5% (distance from the exact Nash equilibrium). Results in column "Maximum" may seems weak, but only in a few examples such result was found. In the Section 2 we have mentioned a few algorithms, which are capable to compute the worst case approximate Nash equilibrium, where ϵ is equal to 0.3393 and 0.5.
In the Table 2 we can see percentage of runs (from 50 runs) in which solution better than fixed value (respectively ϵ less than 0.3, 0.2 and 0.1). As we can see, only small number of runs returned value worse than $\epsilon = 0.3$. Opposite situation occurs in the example 9 (Table 2), were only 4% of runs returned solution better than 10%. Games for the experiments was chosen randomly, but some of them may be a little harder than the other. At this point, we can't indicate properties of such difficult game.
Finally, on the Fig. 12 we present quality of solutions for 10 different problems (averaged values from 50 runs). Mainly, this is graphical representation of first table with some added values - first and third quantile. It allows to see that the obtained results are very good, and average ϵ value is near 0.15.

7. Conclusions

The Differential Evolution (DE) was chosen out of all evolutionary algorithms (EAs) available, to speed up the execution, since the DE algorithm is one of the strongest approaches of all evolutionary algorithms. After the analysis of all test sets, the following conclusions may be reached: the Differential Evolutio was capable to solve every game (Table 1), and average ϵ was close to 0.15. Unlike mathematical methods, the proposed approach is capable to give different solutions in every run of the algorithm.

Table 1. Average, minimum, maximum, mean and standard deviation ϵ values for 10 different tested problems.

Game number	Average ϵ	Minimum ϵ	Maximum ϵ	Mean	Standard deviation
Game #1	16.5%	2.9%	35.4%	13.8%	8.3%
Game #2	18.8%	3.2%	39.0%	18.4%	10.1%
Game #3	19.0%	3.1%	42.4%	19.2%	9.7%
Game #4	14.8%	2.3%	36.6%	12.4%	7.4%
Game #5	13.3%	3.3%	46.1%	12.0%	8.7%
Game #6	15.1%	4.2%	38.4%	13.5%	8.0%
Game #7	18.3%	4.9%	45.4%	15.0%	9.4%
Game #8	13.8%	2.4%	32.8%	11.9%	6.8%
Game #9	20.5%	3.5%	35.4%	21.0%	7.3%
Game #10	15.8%	2.1%	41.5%	13.9%	10.5%

Table 2. Percentage of runs, in which solution was better than fixed ϵ values equal to 0.3, 0.2 and 0.1.

Game number	$\epsilon < 0.3$	$\epsilon < 0.2$	$\epsilon < 0.1$
Game #1	94%	68%	30%
Game #2	78%	62%	24%
Game #3	88%	56%	24%
Game #4	96%	78%	20%
Game #5	94%	90%	38%
Game #6	94%	70%	24%
Game #7	90%	52%	16%
Game #8	94%	84%	36%
Game #9	92%	40%	4%
Game #10	90%	62%	36%

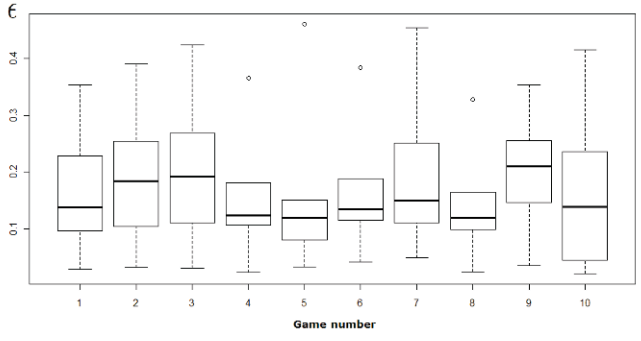

Figure 12. Quality of solutions for 10 different problems.

In this article we proposed and tested a new method for calculating ϵ-Nash equilibria in the bimatrix, non-zero sum games. The new approach seems to be a good and simple alternative for existing solutions. Only for a few examples, reached solutions were not satisfactory.

8. Future works

Our main goal is to develop a method which could be clearly compared with other algorithms for computing Nash equilibria. At this stage of the research, we where able to transform the problem to the clasical function optimization problem. This allows us to focus on the fitness function development. In this article we presented only results concerning the basic differential evolution. Especially interesting seems to be comparison of different mutation schemas.
One of our next goals is to determine properties of games which are difficult to solve for the differential evolution algorithm and modified approach. Of course, our main goal is to expand the scope of the proposed algorithm to the whole class of non-zero sum random games. At this point, games with known support seems to be a serious limitation.
All the considerations involved transforming the strategic form game into function optimization problem. The differential evolutiom seems to be a very flexible algorithm. For example, special geometric operators were developed for discrete problems. It's worth to investigate whether it is possible to create mutation operator directed for the presented problem.

References

[1] Aubin J., Mathematical Methods of Game and Economic Theory (North-Holland Publ. CO., New York, 1979)
[2] Boryczka U., Juszczuk P., Klosowicz L., A Comparative Study of Various Strategies in Differential Evolution. KAEiOG 2009 - Evolutionary Computing and Global Optimization (Ed. J. Arabas, 19-26, Warszawa, 2009)
[3] Bosse H., Byrka J., Markakis E., New Algorithms for Approximate Nash Equilibria in Bimatrix Games, J. Theor. Comp. Sci., 411, 164-173, 2010
[4] Chen X., Deng X., Settling the complexity of two-player nash equilibrium, In: Proceedings of 47th FOCS, 261-272, 2006
[5] Daskalakis C., Goldberg P., Papadimitriou C., The complexity of computing a nash equilibrium, In: Proceedings of STOC, 71-78, 2006
[6] Daskalakis C., Mehta A., Papadimitriou C., A note on approximate Nash equilibria, Workshop on Internet and Network Economics, 297-306, 2006
[7] Daskalakis C., Mehta A., Papadimitriou C., Progress on approximate Nash equilibria. In: ACM Conference on Electronic Commerce, 355-358, 2007
[8] Daskalakis C., Goldberg P.W., Papadimitriou C., The complexity of computing a Nash equilibrium, Mag. Comm. ACM, 52, 89-97, 2009
[9] Froelich W., Juszczuk P., Predictive Capabilities of Adaptive and Evolutionary Fuzzy Cognitive Maps - A Comparative Study, Studies in Comput. Int., 252, 153-174, 2009
[10] Hammerstein P., Selten R., Handbook of game theory - Chapter Game theory and evolutionary biology (University of Bonn, 1994)
[11] Jun H., Chun G., An Electronic Marketplace Based on Game Theory, In: Proceedings of the 2009 Third International Symposium on Intelligent Information Technology Application, 3, pp. 111-114, 2009
[12] Kaplan T., Dickhaut J., A Program for Finding Nash Equilibria (University of Minnesota, Department of Economics in its series Working papers, number 004)
[13] Krohn I., Moltzahn S., Rosenmuller J., Sudholter P., Wallmeier H.M., Implementing the modified LH algorithm, Appl. Math. Comput., 45, 31-72, 1991
[14] Lampinen J., Zelinka I., Mixed variable non-linear optimization by differential evolution, In: Proceedings of Nostradamus, 45-55, 1999
[15] Lampinen J., Zelinka I., On stagnation of the differential evolution algorithm, In: Proceedings of Mendel, 6th International Mendel Conference on Soft Computing, 76-83, 2000
[16] Lemke C.E., Howson J.T., Equilibrium Points of Bimatrix Games, J. Soc. Ind. Applied Math., 12, 413-423, 1964
[17] Li Y. et al., Image Reconstruction of EIT using Differential Evolution Algorithm, In: Proceedings of the Twenty - Fifth IEEE Annual International Conference on Engineering in Medicine and Biology Society, 2, 1011-1014, 2003
[18] Magoulas G.D., Plagianakos V.P., Vrahatis M.N., Neural Network- BasedColonoscopic Diagnosis using On-Line Learning and Differential Evolution, Appl. Soft. Comput., 4, 369-379, 2004

[19] Richard D. McKelvey, Andrew M. McLennan, Theodore L. Turocy, Gambit: Software Tools for Game Theory, Version 0.2010.09.01. http://www.gambit-project.org, 2010
[20] Milchtaich I., Computation of completely mixed equilibrium payoffs in bimatrix games, Int. Game Theory Rev., 8, 483-487, 2006
[21] Nash J.F., Non-cooperative games, Ann. Math., 54, 286-295, 1951
[22] Nudelman E., Wortman J., Shoham Y., Leyton-Brown K., Run the GAMUT: A Comprehensive Approach to Evaluating Game-Theoretic Algorithms, In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2, 2004
[23] Ordeshook P., Game theory and political theory (Cambridge University Press, 1986)
[24] Panagopoulou P., Spirakis P., Approximate and well-supported approximate Nash equilibria of random bimatrix games, In: Proceedings of the 11th Panhellenic Conference on Informatics, 56-578, 2007
[25] Paterlini S., Krink T., High Performance Clustering with Differential Evolution, In: Proceedings of the IEEE Congress on Evolutionary Computation, 2, 2004-2011, 2004
[26] Porter R., Nudelman E., Shoham Y., Simple search methods for finding a Nash equilibrium, Games and Economic Behavior, 664-669, 2004
[27] Price K., Storn R., Lampinen J., Differential evolution: a practical approach to global optimization (Springer Verlag, 2005)
[28] Sandholm T., Gilpin A., Conitzer V., Mixed-integer programming methods for finding Nash equilibria, In: Proceedings of the 20th national conference on Artificial intelligence, 2, 495-501, 2005
[29] Spirakis P., Tsaknakis H., An optimization approach for approximate Nash equilibria. In: WINE'07 Proceedings of the 3rd international conference on Internet and network economics, 42-56, 2007
[30] Storn R., Price K., Differential evolution - a simple and effcient heuristic for global optimization over continuous spaces, J. Glob. Optimiz., 11, 341-359, 1997
[31] Sureka A., Wurman P., Using tabu best-response search to find pure strategy nash equilibria in normal form games, In: 4th Int. joint conference on Autonomous agents and multiagent systems, 1023-1029, 2005
[32] Tennenholtz M., Game Theory and Artificial Intelligence, Springer Verlag, 2403, 34-52, 2002
[33] Tsaknakis H., Spirakis P., A Graph Spectral Approach for Computing Approximate Nash Equilibria, ArXiv e-prints, 96-96, 2009
[34] Widger J., Grosu D., Computing Equilibria in Bimatrix Games by Parallel Support Enumeration, International Symposium on Parallel and Distributed Computing, 250-256, 2008