HGSNash Evolutionary Strategy as an Effective Method of Detecting the Nash Equilibria in n-Person Non-Cooperative Games

Krzysztof Jauernig[1] and Joanna Kołodziej[2] and Michał Stysło

University of Bielsko - Biała, Department of Mathematics and Computer Science,
Bielsko-Biała, Poland
[1] email: kjauernig@ath.bielsko.pl [2] email: jkolodziej@ath.bielsko.pl

Abstract. We present a parallel hierarchical evolutionary strategy HGSNash as a new method of detecting the Nash equilibria in n-person non-cooperative games. The problem of finding the equilibrium points is formulated as a global optimization problem. A definition of the strategy and results of some simple numerical experiments are also included.

1 Introduction

Game Theory can be understood as a mathematical theory for making decisions in conflicts among decision-makers, called *players*. A *game* is a model of strategic interactions among a number of players. Each player tries to choose the *strategy*, which maximizes his utility in taking part in the game. The end result of the game depends on the strategies chosen by each player. A *solution* of the game is a systematic description of the outcomes that may appear in a game. The most commonly encountered solution concept in game theory is the *Nash equilibrium* . The problem of detecting the Nash equilibria of a finite strategic game remains a challenging problem. It can be also formulated as a global optimization problem (see [5]). In this paper we propose a new parallel evolutionary strategy HGSNash, especially designed for Nash equilibria detection. It is based on the HGS algorithm (see [3]), which is a very effective tool in solving ill-posed optimization problems.

The remainder of the paper is organized as follows. Sections 2 and 3 introduce notions and state the problem. In section 4 we define the HGSNash strategy. Results of performed numerical experiments are reported in Section 5. The paper ends with some final remarks.

2 Nash equilibrium in n -person games

Let us assume that n players ($n \in \mathbb{N}$) take part in a game.

Definition 2.1. An n - person game G_n can be defined by the tuple $(N, \{S_i\}_{i\in N}, \{Q_i\}_{i\in N})$ where:

- $N = \{1, ..., n\}$ is the set of players,
- $\{S_1, \ldots, S_n\}$ ($\sharp S_i \geq 2; i = 1, \ldots, n$) is the set of strategies for the players,

- $\{u_1, \ldots, u_n\}; u_i : S_1 \times \cdots \times S_n \rightarrow \mathbb{R}; \forall_{i \in N}$ is the set of payoff functions of the players.

The set $S(N) \subset S_1 \times \cdots \times S_n$ denotes the set of the strategies combinations and is called *a feasible multi strategy* for the game. Sometimes the players can cooperate. In this case we can speak of *a cooperative game* as opposed to *a non-cooperative game*, in which players are not allowed to cooperate. Every player tries to minimize their payoff function during the game. The most commonly encountered concept of the game solution is an equilibrium point defined below.

Definition 2.2. An n-dimensional vector $(\bar{s}_1, \ldots, \bar{s}_n)$ of strategies is called **an equilibrium point** or **Nash equilibrium**, if :

$$\forall_{i \in N} \quad u_i(\bar{s}_1, \ldots, \bar{s}_n) = \min_{s_i \in S_i} u_i(\bar{s}_1, \ldots, \bar{s}_{i-1}, s_i, \bar{s}_{i+1}, \ldots, \bar{s}_n) \tag{1}$$

The Nash equilibrium can be interpreted as a steady state of the play of a strategic game, in which each player holds correct expectations concerning the other players behavior. If the strategies chosen by all players are Nash equilibrium, no player is interested in changing their strategy, because of the decreasing value of their payoff function.

An n-vector $\bar{u} = (u_1(\bar{s}_1, \ldots, \bar{s}_n), \ldots, u_n(\bar{s}_1, \ldots, \bar{s}_n))$ is called *a value* of the game and the strategies $(\bar{s}_1, \ldots, \bar{s}_n)$ are called *pure* strategies. It means that they are never changed during the game.

In this paper we consider only games with non-zero sums for which Nash points are the results of the minimization of a multi-loss function. We also consider only these Nash points which are not on the boundary of the admissible domain.

The problem of detecting the Nash equilibria of a finite strategic non-cooperative game can be also formulated as a global optimization problem.

Let us define a set of loss (cost) functions for the players:

$$\{Q_1, \ldots, Q_n\}; Q_i : S_1 \times \cdots \times S_n \rightarrow \mathbb{R}; \forall_{i=1,\ldots,n} \tag{2}$$

Let us also define a set of *players' response functions* $r_i : S_1 \times \cdots \times S_n \rightarrow \mathbb{R}$ in the following way:

$$\forall_{i \in N} \quad r_i(\hat{s_i}) = \min_{s_i \in S_i} \{Q_i(s_1, ..., s_n)\}, \tag{3}$$

where $\hat{s_i} = (s_1, ..., s_{i-1}, s_{i+1}, ..., s_n)$. The response function defines the optimal strategy for the player.

We can now define *a multi-loss function* $Q : S_1 \times \cdots \times S_N \rightarrow \mathbb{R}$ for the game by the following formula:

$$Q(s_1, ..., s_n) = \sum_{i=1}^{n} [Q_i(s_1, ..., s_n) - \min_{s_i \in S_i} Q_i(s_1, ..., s_n)] \tag{4}$$

Note that the multi-loss function has non-negative values. It can be proved (see[2],[5],[8]) that the Nash equilibrium is the result of the global minimization of the function Q. The players' strategies are called the decision variables and the players' loss functions are called players' objective functions.

It follows from the definition of the function Q that we need to minimize first the loss functions of the players and then we can compute the values of the multi-loss function. Thus the procedure of detection of the Nash equilibria must be a parallel algorithm composed of two cooperated units:

- **Main unit** - which solves the problem of the global minimization of the Q function,
- **Subordinate unit** - which solves the problems of the minimization of the players' loss functions Q_i.

The "Subordinate unit" could be a parallel algorithm designed for the numerical optimization of the real functions of several variables.

The computation of the gradient of the multi-loss function given by the formula (4) is usually very hard or impossible. The non-gradient global optimization algorithms are recommended as the main unit algorithms (for example Powell algorithm or Hooke-Jeeves algorithm). The non-gradient methods can be also applied for the minimization of the players loss functions in the subordinate unit. The proposition of such parallel algorithms are given by Ślepowrońska in [8]. She applied the non-gradient method of the exterior penalty function or Advanced Controlled Random Search algorithm as the main units algorithms. She also used in the subordinate unit non-gradient global optimization algorithms : Hook-Jeeves and Powell methods. But in last few years genetic algorithms have also become very popular. One of the most effective algorithms is the Hierarchical Genetic Strategy (HGS).

3 HGSNash algorithm

3.1 Hierarchical Genetic Strategy

Hierarchical Genetic Strategy (HGS), introduced by Kołodziej et al (see [4]), is a kind of multideme, parallel evolutionary algorithm, which is a very effective tool in solving ill-posed global optimization problems with multimodal and weakly convex objective functions. High efficiency of the strategy comes from the concurrent search in the optimization landscape by many small populations. The sequences of these populations are defined as the evolutionary dependent processes.

The dependency relation among processes has a tree structure with a restricted number of levels. The processes of lower order represent chaotic search with low accuracy. They detect the promising regions on the optimization landscape, in which more accurate process of higher order are activated. Every process creates a branch of the tree. Populations evolving in different processes can contain individuals which represent the solution (the phenotype) with different precision. This precision can be achieved by binary genotypes of different length and by different values of mutation parameter.

The strategy starts with the process of the lowest order called root. After the fixed number of evolution epochs the best adapted individual is selected. We call this procedure the *metaepoch* of the fixed period. After every metaepoch a new process of the higher order can be activated. This procedure is called the *sprouting operation*. The detailed definitions of those procedures can be found in [4]. We can consider two different kinds of the implementation of HGS: binary and real. In the binary implementation of HGS (see [3,4]) the individuals have the binary genotypes and the Simple Genetic Algorithm

(SGA) was applied as *the law of evolution* in every process. The replacement of the SGA engine by the simple evolutionary algorithm with Gaussian mutation, simple arithmetic crossover and real coded genotypes increased the efficiency of HGS (see [9]).

3.2 The main idea of HGSNash

The main idea of HGSNash is an adaptation of the Hierarchical Genetic Strategy (HGS) to the global optimization of the multi-loss game function Q defined by the formula (4). This adaptation requires a hybridization of the main evolutionary mechanism in HGS, because of the parallel structure of the multi-loss function optimization procedure.

Phenotypes of the individuals in HGSNash populations are n-dimensional vectors of the decision variables of the players. We can define genotypes of the individuals by coding their phenotypes into binary strings or using their real representation. Thus, as for HGS, we can analyze the binary and real implementations of HGSNash. The basic mechanism of evolution in HGSNash depends on the implementation type. We applied a hybrid simple genetic algorithm in the case of the binary HGSNash implementation and a hybrid simple evolutionary algorithm in the case of its real implementation. Another genetic operators defined originally for HGS, i.e. sprouting operator, prefix comparison operator (see [4] and [9] for details), can be directly implemented also in HGSNash.A k-periodic metaepoch in HGSNash ($k \in \mathbb{N}$) can be defined in the way presented in Figure 2.

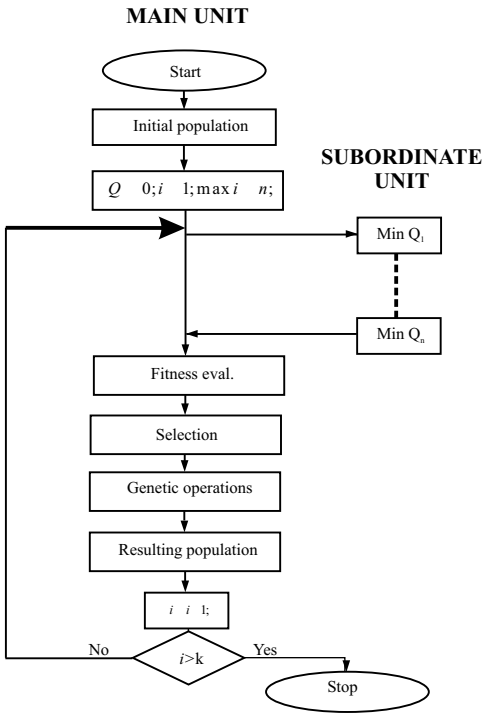

Figure 1. A k-periodic metaepoch in HGSNash

As the subordinate units in HGSNash we propose non-gradient methods because of the problems with gradient computation for a wide class of the objective functions. In the implementation of the method presented in this paper we applied the Powell optimization method (see[8] for details).

4 Experiments

We performed some simple numerical experiments for the verification of the efficiency of HGSNash. We divided our test problems into three groups depending on the computational complexity levels.

4.1 Test problems

Test 1

The goal of the first experiment was the ability of finding the multiple Nash equilibria in case of the very simple 2-person game rules. The players' loss function are defined by the following formulas:

$$\begin{array}{rcl} Q_1(s_1, s_2) & = & (s_1 - s_2 + 1)^2 \\ Q_2(s_1, s_2) & = & (s_2 - s_1^2)^2 + (s_1 - 1)^2 \end{array}$$

where $s_1 \in [-1; 2, 5]$ and $s_2 \in [-1, 3]$.

There exist two Nash equilibria for this game:

$$\begin{array}{rcl} s^{n1} & = & (-0,618033; 0,381966) \\ s^{n2} & = & (1,6180339; 2,6180339) \end{array}$$

Test 2

In this example we also analyze a 2-person game, but each player has 9 decision variables. The players' loss functions are defined below:

$$\begin{array}{rcl} Q_1(s_1, ..., s_{18}) & = & (s_1 - 1)^2 + (s_2 - 1)^2 + s_3^2 + (s_4 - 1)^2 + s_5^2 + (s_6 - 1)^2 + (s_7 - 1)^2 \\ & & +s_8^2 + s_9^2 + s_{11}^2 + (s_{12} - 0,5)^2 + s_{13}^2 + (s_{16} + 0,5)^2 + (s_{18} - 1)^2 \\ Q_2(s_1, ..., s_{18}) & = & (s_{10} + 1)^2 + s_{11}^2 + (s_{12} - 1)^2 + s_{13}^2 + s_{14}^2 + (s_{15} + 1)^2 + (s_{17} - 1)^2 \\ & & +s_{16}^2 + (s_{18} - 1)^2 + (s_2 - 0,5)^2 + s_3^2 + (s_4 - 0,5)^2 + (s_8 - 0,5)^2 \end{array}$$

where $s_i \in [-2, 2]$, forall $i = 1, ..., 18$. A solution of this game is the unique Nash equilibrium:

$$s^n = (1, 1, 0, -1, 0, 1, 1, 0, 0, -1, 0, 1, 0, 0, -1, 0, 1, 1).$$

4.2 Parameters of the strategy

For solving the problems defined in the previous section we applied 3-levels HGSnash strategy in both binary and real implementations. Binary coded individuals have genotypes of different lengths. The genotypes were elongated in populations on the higher

Table 1. Values of parameters for the binary inplementation of 3- levels HGSNash

Parameter	**Level1**	**Level2**	**Level3**
Population size	20	10	10
Binary code length	6	12	18
Mutation probability	0,1	0,05	0,025
Crossover probability	0,9	0,9	0,9
Metaepoch period	10	10	10

Table 2. Values of parameters for the real inplementation of 3- levels HGSNash

Parameter	**Level1**	**Level2**	**Level3**
Population size	20	10	10
Mutation parameter	1	0,5	0,25
Metaepoch period	10	10	10

levels while the sizes of these populations and the mutation rate were decreased. The values of all input parameters for the binary implementation of HGSNash are presented in Table 1.

In the real implementation of HGSNash we applied the Gaussian mutation operator with the standard deviation defined as *a mutation parameter*. The values of parameters set for the 3-levels strategy are sampled in Table 2.

In both implementations of HGSNash we applied as the subordinate unit the Powell optimization algorithm. As the stop criterion for our strategies we accepted the maximal number of metaepochs executed in the single run of the algorithm, which was 200 in every experiment.

4.3 Results of the experiments

Every experiment was repeated 30 times. Table 3 reports the number of runs in which the global optimum was found (nr) along with the average objective values of the best individuals found in the end of each run (avg). The real implementation of HGSNash was better in finding the solutions of all problems. We also wanted to know how quick are the strategies in detecting the Nash equilibria. Analyzing Table 4 we can see how many metaepochs were needed for finding all the solutions of the test problems. We expected that the real implementation of HGSNash should be more effective than the binary one. We were surprised that it holds only for the first test. Is it true that the real HGSNash was quicker only for the simplest problem? To answer this question we computed the average execution times of the algorithms and obtained the sample results in Table 5.

The results show that binary coding, mutation and the crossover of the individuals in binary HGSNash are very time consuming operations. The strategy was up to 8 times

Table 3. The values of (nr) and (avg) after 200 metaepochs

Strategy	**Test1** nr	**Test1** avg	**Test2** nr	**Test2** avg
HGSNash binary	20	0,05	17	0,1
HGSNash real	30	0,006	25	0,09

Table 4. The average number of metaepochs needed for finding the Nash equilibria

Strategy	**Test1**	**Test2**
HGSNash binary	21	16
HGSNash real	7	35

Table 5. The average time of the single run of the strategy (in seconds)

Strategy	**Test1**	**Test2**
HGSNash binary	8	14
HGSNash real	1	3

slower than its real implementation. In the end we compared the results obtained for the both HGSNash implementations with the results of the similar experiments performed for the parallel algorithms ARCS and HJ defined in [8].

One iteration of the ARCS and HJ algorithm was equivalent to the single genetic period of metaepoch in HGSNash. Only in the third test are the results on the same level. Evolutionary strategies were much more effective in solving 2-person games.

Table 6. The average number of iterations needed for finding the Nash equilibria

Strategy	**Test1**	**Test2**
HGSNash binary	210	160
HGSNash real	100	350
ARCS	687	24398
HJ	468	4378

5 Conclusions

- Finding Nash equilibria of strategic games can be difficult and tedious. Algorithms for solving games have been studied for the beginning of game theory but usually they are very time consuming.

- the HGSNash strategy presented in this paper is an evolutionary optimization method based on the Hierarchical Genetic Strategy.

- HGSNash is very effective in solving 2-persons games with very complicated rules and it is not worse than other parallel algorithms in solving n-person games with multiple Nash equilibria.

- We would like to find some decision-making problems for the practical application of HGSNash.

Bibliography

[1] Arabas J.: *Lectures on the Evolutionary Algorithms (in Polish)*, WNT, Warszawa 2001.

[2] Ewald, C.: *Games, Fixed Points and Mathematical Economics.* University of Kaiserslautern 2003/2004.

[3] Kołodziej J.: Hierarchical genetic Strategy as a New Method in Parallel Evolutionary Computation, *Proc. of the 2-nd International Conf. on Formal Methods and Intell. Techn. in Control, Decision, Multimedia and Robotics*, Polish-Japanese Institute of Inf. Technology, Warszawa, s. 50-58.

[4] Kołodziej J., Gwizdała R., Wojtusiak J.: Hierarchical Genetic Strategy as a Method of Improving Search Efficiency, *Advances in Multi-Agent Systems*, R. Schaefer and S. Sędziwy (Eds.), UJ Press, Krakw, Chapter 9, s. 149-161.

[5] Pavlidis N.G. et al: Computing Nash Equilibria Through Computational Intelligence Methods, *J. of Computational and Appl. Math.*, 175 (2005), pp.113-136.

[6] Schaefer R., Kołodziej J.: Genetic Search Reinforced by The Population Hierarchy, *FOGA VII*, Morgan Kaufmann, pp. 383-401.

[7] Straffin, P.: *Game Theory.* Scholar Press (Polish ed.), Warszawa 2004.

[8] Ślepowrońska K.: A parallel algorithm for the Nash equilibria detection (in Polish) *MSc Thesis*. Warsaw Technical University Press, Warszawa 1996.

[9] Wierzba B., Semczuk A., Kołodziej J. and Schaefer R.: Hierarchical Genetic Strategy with real number encoding. *Proc. of KAEiOG'03.* Łagów Lubuski, 26-28.05.2003, pp.231-239.