EVOLVING STRATEGIES: A NUMERICAL APPROACH TO GAME-THEORETIC EQUILIBRIUM ANALYSIS

Rykunov D.P. (Russian Federation)

Rykunov Dmitrii Pavlovich – BSc in Economics,
FACULTY OF ECONOMICS,
NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS,
MOSCOW

Abstract: *this article investigates a general-purpose framework to numerically analyze game theoretic models in terms of the stability of equilibria and individual decision-maker behavior. It focuses on models equivalent to finite non-cooperative games. We start with a mathematical formalization of such models and their equilibria and outline an approach to analyze them, based on evolutionary game theory concepts. We proceed by implementing the approach in an algorithm inspired by evolutionary optimization algorithms, capable of locating multiple equilibria with an iterated local search procedure. The algorithm's convergence to equilibria concepts traditional for evolutionary game theory studies, namely evolutionary stable strategy and evolutionary stable set, is demonstrated by empirical results.*

Keywords: *game theory, evolutionary algorithms, Nash equilibrium, prisoner's dilemma, computational modeling, strategic interactions, multiple equilibria, genetic algorithm, evolutionary game theory, algorithmic game theory, agent-based modeling.*

РАЗВИВАЮЩИЕСЯ СТРАТЕГИИ: ЧИСЛЕННЫЙ ПОДХОД К ТЕОРЕТИКО-ИГРОВОМУ АНАЛИЗУ РАВНОВЕСИЯ

Рыкунов Д.П. (Российская Федерация)

Рыкунов Дмитрий Павлович – бакалавр экономики,
Факультет экономики,
Национальный исследовательский университет «высшая школа экономики»,
г. Москва

Аннотация: *в этой статье исследуется универсальная структура для численного анализа теоретико-игровых моделей с точки зрения устойчивости равновесий и поведения отдельных лиц, принимающих решения. Основное внимание уделяется моделям, эквивалентным конечным некооперативным играм. Мы начнем с математической формализации таких моделей и их равновесий и наметим подход к их анализу, основанный на концепциях эволюционной теории игр. Мы продолжим реализацию подхода в алгоритме, вдохновленном алгоритмами эволюционной оптимизации, способном находить множественные равновесия с помощью повторяющейся процедуры локального поиска. Эмпирические результаты демонстрируют сходимость алгоритма к традиционным для исследований эволюционной теории игр понятиям равновесия, а именно к эволюционно устойчивой стратегии и эволюционно устойчивому множеству.*

Ключевые слова: *теория игр, эволюционные алгоритмы, равновесие Нэша, дилемма заключённого, компьютерное моделирование, Стратегические взаимодействия, множественные равновесия, генетический алгоритм, эволюционная теория игр, алгоритмическая теория игр, агентное моделирование.*

UDC 330.42

1. Introduction

In the realm of economic and social sciences, understanding the dynamics of strategic interactions is paramount. The most fundamental proposition in this context is a Nash Equilibrium concept, based on an assumption of mutually known full rationality of decision-making agents; it proved itself as a useful concept for theoretic analysis of social interactions. Despite its widespread application, the Nash Equilibrium presents a significant challenge: the potential for multiple equilibria, which complicates the prediction of outcome scenarios in strategic interactions.

In this paper, we propose that if there are multiple equilibria in a model of social phenomena being investigated, then it is desirable to be able to make plausible predictions of its dynamics and have an opportunity to influence it.

Tackling the problem of locating multiple equilibria of game-theoretic models could be started with a simpler task of locating at least one equilibria. A lot of work was done on creating analytical approaches to calculate equilibrium points [1] [2] [3] [4] [5] [6], [7]. In addition, there are prior works on the application of

evolutionary game theory and evolutionary optimization algorithms to the same task [8] [9] [10] [11] [12] [13] [14]. There are also some works on general-purpose genetic algorithms using concepts from game theory, called Nash Genetic Algorithms, these algorithms are primarily applied to solving engineering problems [15].

Algorithms locating all Nash Equilibria in a given game are constructed around the naïve idea of utilizing a formal definition of NE and using systems of polynomial equalities and inequalities based on the game's payoff structure. There is a major disadvantage of such an approach: such systems of polynomial inequalities scale up fast with game size (number of players, strategies involved), quickly becoming infeasible to solve [2].

In this paper, we focus on using an evolutionary optimization approach to tackle this problem. There is a long history of applications of evolutionary algorithms in the field of Game Theory [14]. Starting with the works of Axelrod on repeated prisoners' dilemma [16]. In addition, some works confirm efficiency of evolutionary optimization algorithms in the task of search for equilibria [9] [11] [10] [13] [8]. These works utilize a coevolutionary approach and even though the proposed algorithms' performance hugely depends on parameters being used, meta-optimization of these parameters is generally not being done. Moreover, the performance of these algorithms is often measured in non-comparable statistics, such as the number of iterations needed for convergence, that could not be directly related to time or computational resources usage, therefore we incorporate in our work empirical analysis of the influence of parameters of the proposed algorithm on its convergence dynamic.

2. Model and Algorithm Specification

In the paper, we try to come up with a general-purpose algorithm to investigate the dynamic of social models taking into account the possible multiplicity of equilibria. Therefore, we want our algorithm to possess the following properties: flexible model specification, the capability to handle multiple equilibria, and insight into the dynamics of the model. To do this we start up with a formalized definition of social interaction or game in which we are going to search for equilibria. Then we define what would be treated as equilibria or stable state of the model and show how a model could converge to one or another equilibrium from an arbitrary state. After that, we propose a numeric algorithm to tackle this model, inspired by evolutionary optimization algorithms and possessing all the desired properties listed before.

Model of Social Interaction

We focus our research on non-cooperative game theoretic models, though the analysis can be naturally extended to other model formulations. The general structure of a game consists of:

- a set of agents
- for each agent, a set of actions
- payoff function that assigns a payoff to each agent

$$\Gamma = (\mathrm{A}, \Delta, \Psi)$$

Γ - game
A - agents set
Δ - actions set
Ψ - payoff function

The agent set A is a finite set of size n that serves as a list of agents being considered in the model.

$$\mathrm{A} = \{a_1, \dots, a_n\}, |\mathrm{A}| = n$$

Actions set Δ is a space of all possible actions that agents could undertake in the model. Action sets could be divided into subsets corresponding one-to-one to agents – agents' decision spaces, representing all actions available to a given player. Then each decision space is further broken down into decision sets that could be either discrete or continuous choices, representing for a given agent available choices in a particular decision they make.

For example, in some model there could be 10 firms (agents) each having 2 decision sets: first – what product out of 5 possible products to produce (discrete decision-set of 5 elements), second – what proportion of taxes should it pay to government (continuous compact decision-set). In this case, the action set of the model would consist of 10 decision spaces (one for each agent) each including 2 decision sets – discrete and continuous – 20 decision sets in total.

$$\Delta = \{D_1, \dots, D_n\},$$

D_i - decision space;

$$\Delta \cong \mathrm{A};\ |\Delta| = |\mathrm{A}| = n;$$

$$D_i \neq \emptyset, \qquad D_i = \{d_{i_1}, \dots, d_{i_m}\};$$

d_{i_j} - decision-set

$$\forall d_{i_j}: d_{i_j}\ is\ a\ discrete\ set\ OR\ a\ compact\ space$$

Payoff function Ψ takes a point from the Cartesian product of decision sets contained in the action set and returns a vector of payoffs of the same cardinality as the agent set, e.g. containing payoff for each agent. Every payoff of an agent is a real value.

$$\Psi: \prod_{d \in \Delta} d \to P;$$
$$P \cong \mathrm{A};\ |P| = |\mathrm{A}| = n;$$
$$P = \{p_1, \dots, p_n\}, p_i \in \mathbb{R}$$

It is assumed that agents have simple preferences over payoffs and payoffs satisfy four axioms of Von Neumann–Morgenstern utility theorem [17], thus $p_i \prec p_i' \Leftrightarrow p_i < p_i'$.

Equilibrium Definition

We use the Δ_i notation to denote the subset of the action set that consists of actions related to agent i, and Δ_{-i} notation to denote the subset of the action set consisting of actions of all other agents. This notation is used with other sets in the same manner, where it makes sense.

In our model, we assume that agents could choose their actions stochastically. We represent such rules for each agent by a set of probability distributions over decision sets of the agent, each of which is called a strategy. By doing this, we allow mixed strategies in our model.

$\forall d \in \Delta, \exists \phi$: ϕ is a random variable with probability distribution over d:
$\forall \phi: \phi \in \Phi$
$\Phi_i \coloneqq set\ of\ \phi\ corresponding\ to\ d \in D_i, |\Phi_i| = |D_i|$

We propose that agents want to maximize their expected payoffs by tweaking their strategies. Therefore, each agent solves the following optimization problem:

$$\forall a_i \in \mathrm{A}: \Phi_i = \arg\max_{\Phi_i} \mathbb{E}[p_i \in \Psi(\Phi_i \times \Delta_{-i})], given\ \Delta_{-i}\ constant.$$

We define a stable state (equilibrium) in the model as a state in which no agent can increase their expected payoff, given others' strategies being stable. At least one such point always exists, because in any game-theoretic model exists a Nash Equilibrium in which this condition holds by definition [18].

$$\forall a \in \mathrm{A}: \exists \Phi_a: \mathbb{E}[p_a \in \Psi(\Phi_a \times \Delta_{-a})] \geq \mathbb{E}[p_a \in \Psi((\Phi_a)' \times \Delta_{-a})]$$

We assume agents to change their strategies by small perturbations with a drift towards strategies yielding higher expected payoff until there would be no possible perturbations yielding higher expected payoff than current strategy yields. In this sense, agents in our model always use totally mixed strategies, because of a non-zero chance for each agent of changing their strategy to a strategy that includes a previously unused action with non-zero probability.

Algorithm Specification

Here we summarize the proposed algorithm in pseudo-code:

1. Take $GAME\_MODEL$ as input with parameters
2. Generate an initial subpopulation of uniformly distributed in search space individuals for each decision-set in the model
3. Evaluate fitness for each individual
4. Apply truncating selection and mutation procedures for each subpopulation
5. Construct a new generation of subpopulations
6. If the termination condition is not met, then go to step 3
7. If equilibrium is found, return found equilibrium
8. Return log of convergence dynamic

Algorithm 1: General Structure of the Algorithm

Algorithm Outline

Here we present an algorithm proposed to locate equilibria of game-theoretic models. We start with a stochastic algorithm of GA (Genetic Algorithm) type (a kind of evolutionary algorithm), capable of locating one equilibrium per run, depending on an initial seed used in the algorithm. For locating multiple equilibria, we use repeated local search, e.g. running the algorithm multiple times with different initial seeds, as a robust naïve approach to multi-modal optimization [19, pp. 59-60].

As an input for the algorithm, we expect a game model as outlined in the Model of Social Interaction section of the paper. We want to highlight the fact that for the algorithm to work it is enough to know just the structure of action set Δ (agents and their decision-sets) and an interface to compute corresponding payoffs (there is no need actually to know how these payoffs are calculated). As an output, we expect a found equilibrium or statement of failing to find one, in either case, the output should contain a log of convergence dynamic, which consists of individuals' strategies in each generation and their fitness. This algorithm could be started several times to find multiple equilibria.

We do not discuss in this article the criteria for determining if an equilibrium is found because it depends on the particular type of model being investigated. Due to the stochastic nature of the algorithm and noise in payoffs, every convergence criteria would be based on heuristics, therefore the most achievable method of determining if the algorithm converged is the manual inspection of its convergence dynamics.

General Evolutionary Algorithm Structure

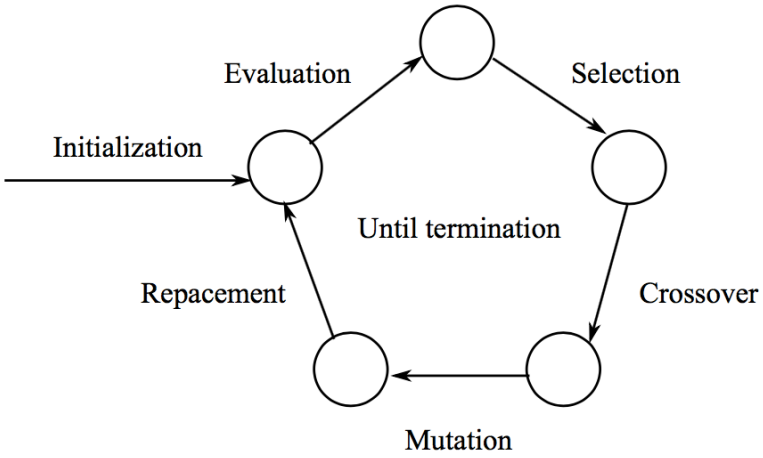


Fig. 1. Flowchart diagram of an arbitrary genetic algorithm. Adapted from (Wei-Kai Lin, 2009, p. 8).

Evolutionary Algorithm (EA) is a large class of optimization routines inspired by the Darwinian evolution process [20, p. 2]. Genetic Algorithm (GA) is a subclass of EA. Its advantage is in universality and it has proved itself as a good-performing search algorithm on many problems. GA uses string representations of solutions in the search space, each such representation called an *individual* or a *species*. A set of such species presented in a given iteration called *population*, iterations called *generation*. To differentiate species in a population *fitness valuation* is used, for example in a task of function maximization a value of a function in a point represented by an individual could be used as a fitness value. The general idea of GA is to evolve populations from generation to generation favoring in transition between generations individuals with the highest fitness value, concurrently maintaining diversity of population. GA routine consists of a few stages; we will use Fig. *1* as a reference for the order of the stages and describe each step presented on the figure clockwise.

1. ***Initialization***. The algorithm is initialized by creating a population uniformly distributed across search space.
2. ***Evaluation***. Each species is assigned a fitness value based on its performance on the target function.
3. ***Selection***. A subset of the population is selected by some rules favoring the highest fit individuals to be used later to produce offspring and/or to be transferred to the next generation.
4. ***Crossover***. Selected individuals produce offspring by a combination of their traits, for example in the case of real-valued vector encoding of points in search space simple averaging of their components might be used.
5. ***Mutation***. Selected individuals (including offspring) change some of their traits in a predefined random manner to explore solution space and maintain the diversity of the population.
6. ***Replacement***. Population of the new generation composed. It is usually preferred to maintain the population size of each generation on the same level, so parents (members of the previous generation) and offspring (potential members of the new generation) should be mixed in some way to form a new generation.

Structure of Populations – Coevolution

Usually, a GA maintains in every generation only one uniform population of individuals representing solutions in a search space. Each species has a full encoding of a solution in the search space. To assign a fitness value to an individual its encoding of a solution is used to calculate the value of the function being optimized and its value is then used to calculate the individual's fitness. This works fine when the function being optimized is single-valued or could be transformed into a single-valued function. However, this is not the case with game-theoretic models where payoffs of agents cannot be aggregated in a meaningful way maintaining independence of strategic interests of agents.

Thus, we encode strategies on different decision sets independently and keep individuals encoding different parts of full solution encoding in distinct subpopulations, therefore we maintain as many subpopulations as decision sets presented in a model. Consequently, a single member of a subpopulation could not be evaluated without matching with some members of other subpopulations, because it is needed to provide a complete point in the Cartesian product of decision sets to the payoff function to derive an individual's payoff and assign fitness. Below we illustrate a possible generation structure of the algorithm dealing with a sequential form of the game "Bach or Stravinsky".

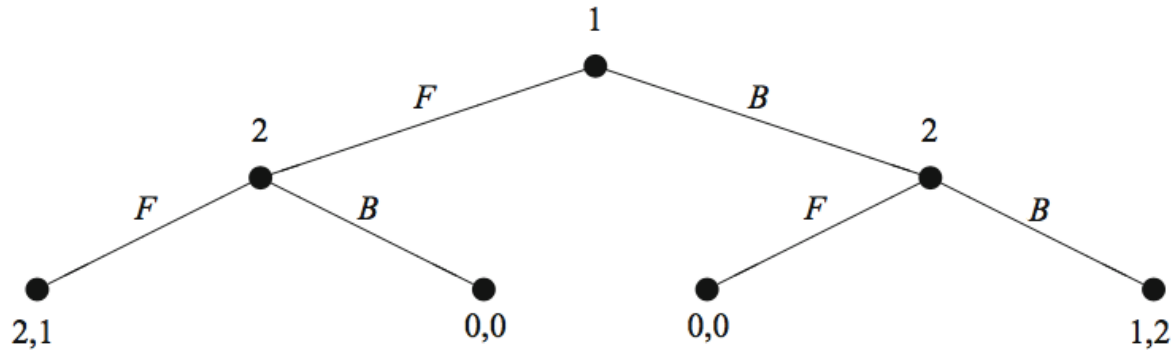


Fig. 2. The decision tree of sequential Bach or Stravinsky. Adopted from [21, p. 8].

Generation ###		
subpopulation 1 **Player 1** **F / B**	**subpopulation 2** **Player 2** **F: F / B**	**subpopulation 3** **Player 2** **B: F / B**
1.000; 0.000	1.000; 0.000	0.000; 1.000
0.991; 0.009	1.000; 0.000	0.003; 0.997
0.975; 0.025	0.989; 0.011	0.000; 1.000
0.992; 0.008	0.995; 0.005	0.000; 1.000

Fig. 3. Example of the structure of a generation of the proposed algorithm for the sequential form of the BoS game.

Fitness Evaluation – Matching Algorithm

To evaluate a species's fitness from a subpopulation its encoding of solution must be completed using encodings of one species out of each other subpopulation. So one needs to match species between subpopulations to evaluate their fitness. Failing to maintain an exchange of encodings between subpopulations results in the algorithm's non-convergence [8].

There are two general approaches to do that in practice: asynchronous and synchronized. In asynchronous evaluation each subpopulation evolves on its own, periodically communicating with other subpopulations to exchange information about reference individuals used to derive complete encodings of solutions [22]. In the synchronized evaluation for each evaluation of an individual of a subpopulation, the individuals from other subpopulations are picked at random [23]. For the implementation of the algorithm in this paper, synchronized evaluation is chosen over asynchronous because of its simplicity and ease of interpretation in the context of fitness value. It should be noted that the actual implementation of the matching algorithm hugely influences the convergence dynamic of an algorithm, as depicted in [24].

We propose a matching algorithm that takes an amount of desired matchings to be produced (same for each individual in the population) as an input $NPAIRS \geq 1$ and returns a list of $NPAIRS * POPSIZE$ matchings (set of individuals, one from each subpopulation, collectively bearing a full encoding of a solution for a model); it is assumed that each subpopulation has the same size of $POPSIZE$. We use $\mathfrak{d}_i$ to denote a subpopulation corresponding to the decision set d_i; a collection of all subpopulations denoted by $\mathfrak{D}$; members of subpopulations are strategies over the corresponding decision-set denoted by ϕ. For reference, check Fig. 3. Example of the structure of a generation of the proposed algorithm for the sequential form of the BoS game**.** Here is an outline of the matching algorithm – Algorithm 2 – in pseudocode:

1. Take input $NPAIRS \geq 1$
2. Initialize list of matchings $\mathbb{M} = \emptyset$
3. Make lists $\mathfrak{d}'$ of members of each subpopulation $\mathfrak{d}$
4. For each member of each subpopulation
5. initialize counter $\rho = 0$
6. Initialize new matching $\mathbb{m} = \emptyset$
7. For each list $\mathfrak{d}'$
8. Take a random individual i from $\mathfrak{d}'$
9. Add i it to the matching $\mathbb{m}$
10. Increment individual's counter by one $\rho_i += 1$
11. If $\rho_i = NPAIRS$, then exclude individual i from list $\mathfrak{d}'$
12. Add matching $\mathbb{m}$ to the list of matching $\mathbb{M}$
13. If any $\mathfrak{d}' \neq \emptyset$, then go to step 5
14. Return list $\mathbb{M}$

Algorithm 2: Matching Algorithm

Finally, to assign fitness values for each matching list, we sample an action for each strategy in the matching, and then we derive payoff function values and increase each matching strategy fitness value by a payoff of the player to which the strategy belongs. This procedure can be repeated several times for the same matching; in the algorithm and further in the paper we refer to the number of times each matching is evaluated as a parameter $NGAMES$. While the algorithm handling mixed strategies, each evaluation of payoff function gives payoffs for sample pure strategies. However, equilibrium or stable states are defined in terms of expected payoffs from mixed strategies. In this context, compounding multiple evaluations of the fitness value of the same strategy is desirable, with the averaged fitness value converging to the true expected payoff as the number of valuations goes to infinity.

Selection – New Generations Formation

For the implementation of our algorithm, we used a truncation selection procedure applied to each subpopulation independently, which in the absence of crossover operations fully determines the dynamics of the construction of new generations. Truncation selection is an algorithm with just one parameter $DROPOUT\_RATE$, which specifies as a proportion $(0; 1)$ of what part of current generation's least fitted individuals would be dropped out and replaced by offspring of more fitted individuals. Selection goes as follows: population members sorted by fitness, then $\lfloor POPSIZE * DROPOUT\_RATE \rfloor$ (but at least one) least fitted individuals are deleted from population and the same amount of offspring of more fitted individuals is produced and added to population.

Offsprings of the most fitted individuals in a population (those left after truncation) are generated by a random selection of individuals in the amount needed to replace dropped individuals, duplicating it and applying mutation operator to duplicates.

Truncation selection is chosen for implementation because of its simplicity and robustness in conjunction with the mutation operator. For the overview of possible selection procedures and discussion of their efficiency please refer to [25].

Mutation

In the proposed algorithm, mutation is the only operation that preserves the diversity of the population and provides search space exploration means for the algorithm. Encodings of solutions in the algorithm consist of real numbers in closed intervals. Hence we construct a mutation operator as a function adding to each component of encoding being mutated a sample of random variable normally distributed with a mean of zero and standard deviation equal to a given percentage of the component's boundary interval length, followed by normalization of encoding to be in alignment with it boundaries. Therefore, the mutation operator also takes just one parameter $MUTATION\_MAGNITUDE$ in a range of $(0; 1]$.

For example, let us consider a case of a strategy over two points discrete decision set being mutated, while MUTATION_MAGNITUDE parameter is set to 0.03. Let us assume that its current encoding is $\{0.5, 0.5\}$. Its encoding consists of two real numbers, so to mutate it we are going to use two samples of random variable with normal distribution with mean 0 and standard deviation $(1 - 0) * 0.03 = 0.03$. So sample obtained and equals to $\{0.184, -0.052\}$. We add it to current encoding and obtain $\{0.684, 0.448\}$ which doesn't sum up to unity $0.684 + 0.448 = 1.132 \neq 1$, thus it cannot be used to encode discrete distribution. To fix that we normalize components of our encoding as follows $\left\{\frac{0.684}{1.132}; \frac{0.448}{1.132}\right\} = \{0.604; 0.396\}$, which sums up to 1 and can be used to encode a discrete probability distribution.

3. Algorithm Assessment

In this chapter, we are going to investigate the performance and convergence of the proposed algorithm empirically by assessing its performance on selected test problems.

Setup of the Experiment

We will be benchmarking our algorithm on a well-known game – "Prisoners' Dilemma" Fig. *4*.

		Player 2	
		Cooperate	**Defect**
Player 1	**Cooperate**	-1; -1	-3; 0
	Defect	0; -3	-2; -2

Fig. 4. Prisoners' Dilemma payoffs matrix.

We want to investigate the influence of parameters on the convergence of the algorithm. To do that we will construct a mesh grid of parameters and evaluate each combination of them several times.

Let us remember that the algorithm depends on 5 parameters: $POPSIZE$, $NPAIRS$, $NGAMES$, $DROPOUT\_RATE$, $MUTATION\_MAGNITUDE$. We construct a mesh grid by taking equally distanced points in specified ranges for each parameter independently and then combining them. We use the following parameters to construct the mesh grid (notice that parameters $dropout\_rate$ and $mutation\_magnitude$ represented here as percentages, rather than fractions):

Notation: 'PARAMETER_NAME':(lower bound, upper bound, number of points)

{'popsize': (5, 100, 7), 'npairs': (1, 40, 7),
'ngames': (1, 1, 1), 'dropout_rate': (1, 80, 12),
'mutation_magnitude': (1, 80, 12)}

In total: $7 * 7 * 1 * 12 * 12 = 7056$ combinations

We keep $NGAMES$ at 1 because its purpose is to control the precision of expected payoff calculation, for which it could be substituted by $NPAIRS$ parameter, but it could not be done the other way around, because $NPAIRS$ parameter also influences the diversity of opponents (or matching encodings) used to evaluate the fitness of each individual. That is, with low values of $NPAIRS$, individuals get biased estimates of expected payoffs of their strategies relative to the surrounding population of opponents to estimate expected payoffs. We leave empirical investigation of the effect of $NGAMES$ parameter on algorithm performance for further research.

Knowing that the only equilibrium of the model is in pure strategies, we propose the following convergence metric: the model converged if and only if for two consecutive generations best-fitted individuals of each subpopulation represent the same pure strategies. When the model converges, we terminate further calculations of the next generations (further in the text we refer to this as the termination condition); also, we stop calculations after 40 generations without convergence.

We evaluate each combination of parameters 10 times for 70560 model evaluations in total.

Results

1. Convergence

First, we construct a histogram showing at which generation the algorithm has converged, non-converged, and false-convergence results are summed together and shown in the left-most bar. The number of counts of each outcome is presented in table Fig. *5*.

Model Run Outcome	false_convergence	non_converged	true_equilibrium
Count	62	8747	61749
Percentage	$< 0.1\%$	12.4%	87.5%

Fig. 5. Results of test runs on Prisoners' Dilemma.

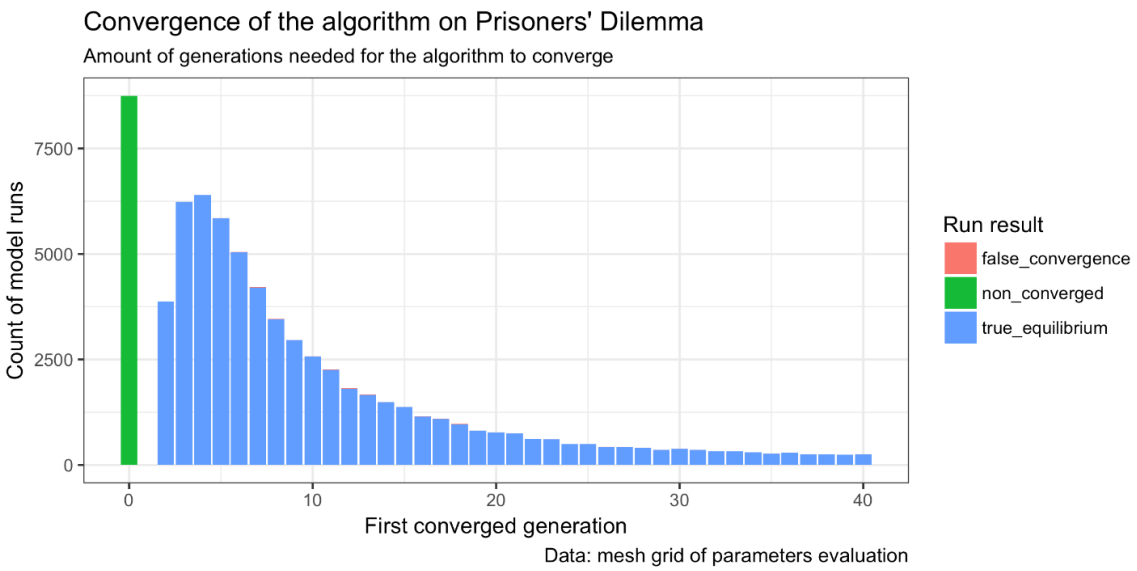


Fig. 6. Convergence of the algorithm on Prisoners' Dilemma problem.

As could be seen from the graph Fig. *6*, using the provided grid of meta-parameters, the model successfully converged in most cases. Also, the number of experiments that resulted in convergence to non-legible equilibrium (false-positive convergence) is negligible.

2. Running Time

To foster faster convergence through exploitation of the parallel nature of the algorithm we increase the number of model evaluations in each generation to provide better guidance to the selection operator and thus increase selection pressure on inferior individuals. Increasing the number of fitness evaluations per generation in a parallel manner means utilizing more computational power and the number of processors used. In the context of scarcity of computational resources, it means higher expenses per unit of time. Because of this, we present a graph of the number of generations needed for the model to converge and the corresponding average amount of fitness evaluations per generation in Fig. *7*.

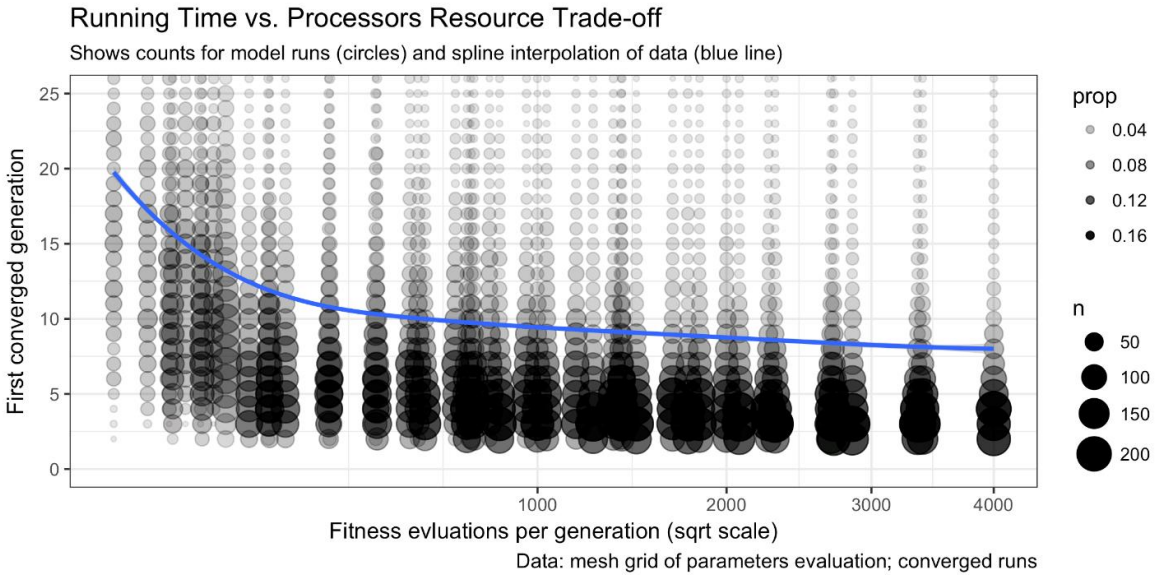
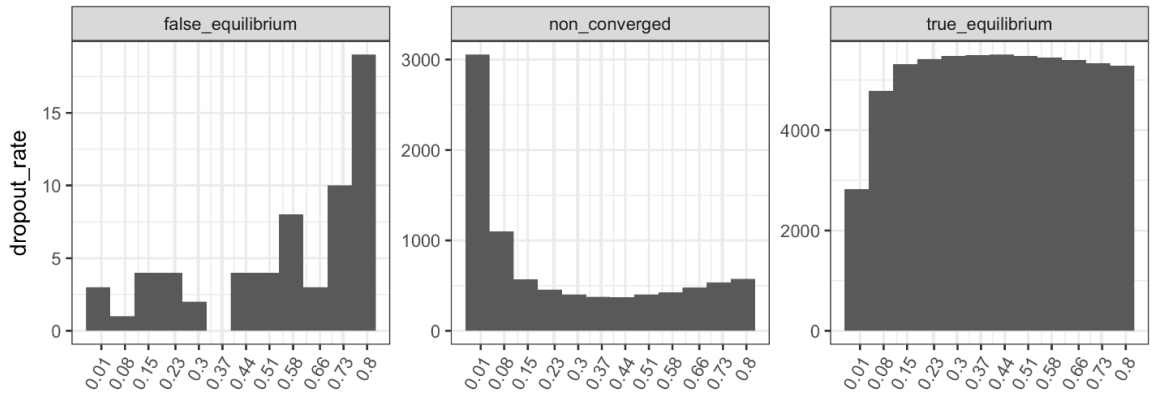


Fig. 7. Running Time vs Processors Resource Trade-off.

As could be seen from the graph, the aforementioned trade-off between running time and computations needed in each generation has an expected form of isoquant. It also could be seen from this graph that the number of fitness evaluations' influence on the number of generations to convergence diminishes quickly after approximately 700 fitness evaluations per generation.

3. Parameters influence

To investigate the influence of the parameters of the algorithm on its performance, we start with investigating parameter distributions among algorithm runs across different outcome categories, showing results in Fig. *8*.



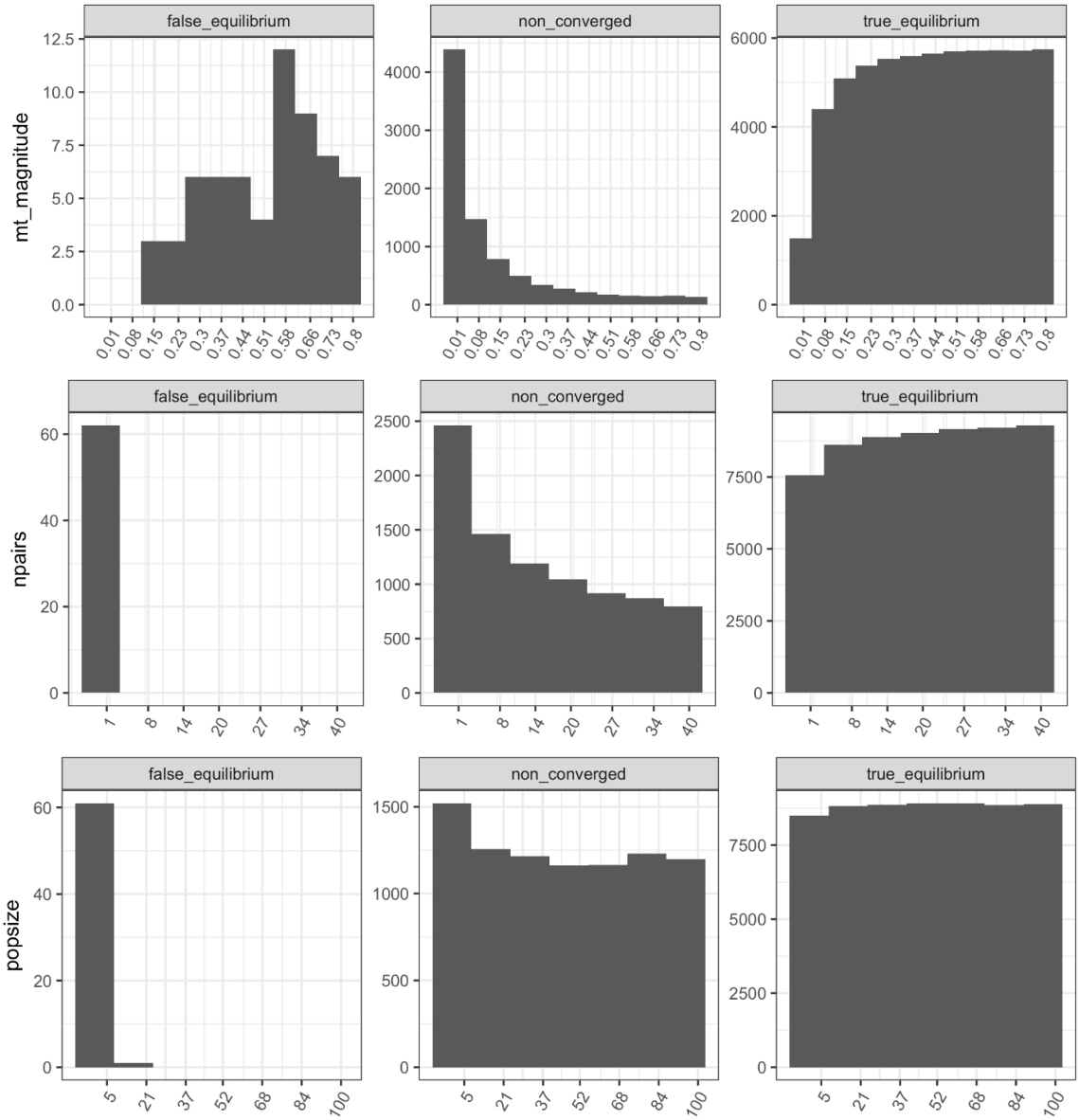


Fig. 8. Parameters' influence on algorithm run's outcome.

The dropout rate at low levels makes the algorithm too conservative in the exploration of search space, thus the algorithm fails to converge in a limited number of generations. It could be said, that with a dropout rate reaching 0, the number of generations needed for the model to converge goes up to infinity – this analysis is valid with an assumption of an infinitely large population because, in the current design of the algorithm, at least one specie is dropped out of population. At the same time, the proportion of runs converging to a non-equilibrium point is rising with an increase in the dropout rate. With high values of this parameter, the algorithm does not preserve enough amount of well-performing species between generations and fails to converge.

Mutation magnitude also defines characteristics of the explorative behavior of the algorithm. It could be expected that the proportion of non-converged models monotonically decreases with an increase in mutation magnitude. The proportion of convergences to non-equilibrium points is generally expected to rise monotonically with mutation magnitude because, with higher values of the parameter, the algorithm traverses search space more sporadically. However, we see different results on empirical data with distribution single peaked around average values of mutation magnitude. It could hypothesized that this effect emerges due to the fact, that higher diversity of population (in terms of variance of strategies) obtained with higher values of the parameters leads to better approximation of expected payoffs of strategies because every one of them is being evaluated against broader range of complementary encodings of solution space.

Expectations of response dynamics of the algorithm on changes of the *NPAIRS* parameter goes fully along with empirical results. With an increase in the parameter's value, the proportion of true equilibria rises, and

false equilibria – declines, non-convergence – declines monotonically. This parameter influences the number of evaluations of each strategy, thus influencing the precision of approximation of expected payoffs of strategies.

The algorithm's response to change in the $NGAMES$ parameter is expected to have the same dynamic as in the case with $NPAIRS$ parameter.

The algorithm's dynamic relative to different sizes of the population is the same as with values of $NPAIRS$ parameter, although the underlying reason is different – the size of the population influences the diversity of the population in terms of amount of maintained species.

Note on Convergence to Evolutionary Stable Strategies

Our algorithm uses truncation selection and mutation as drivers of evolutionary dynamics in the process of convergence. The relation of this selection procedure to the classic replicator dynamic [26] was investigated in [25] and [27]. It was shown there that truncation selection leads an evolutionary system to convergence to an Evolutionary Stable Strategy (ESS).

ESS is a solution concept in evolutionary game theory defined for models allowing only pure strategies for agents as stable proportions of pure-strategist individuals with certain strategies in a population, therefore it could be treated as a stable state of the population as a whole. For now, we restrict our analysis to models with monomorphic ESS, which is ESS that could be represented by a single pure strategy, so by now it is enough to be said that every ESS is a Nash Equilibrium.

The aforementioned works on the convergence of truncation selection to ESS investigate the subject using classic pure-strategist models, in which individuals are restricted to follow just one pre-determined pure strategy throughout their lifecycles. It should be noted that under such conditions, in the absence of diversity-preserving mechanisms, truncation selection under some circumstances might fail to converge, due to extinction of some strategies. However, this is not the case with the algorithm proposed here, since it uses mutation operation to explicitly foster diversity of the population. Hence in the long run it converges asymptotically to an Evolutionary Stable Strategy of processed model (if there is any) by the design.

4. Conclusions and Further Research

In summary, this paper introduces a general-purpose numerical approach to investigate equilibria of mixed-strategy game-theoretic models in practical applications. The conceptual approach is complemented by an algorithmic implementation. The influence of the parameters of the algorithm on its performance is studied. The approach is agnostic to model specification details and treats it as a black box by design. It is also shown and supported by empirical results that the algorithm is convergent to Evolutionary Stable Strategies as defined by [26].

Finally, we suggest the following directions for further research:

- Applications of the algorithm to the analysis of system dynamic and equilibrium states emergence in game-theoretic models
- Analyze the correspondence of the algorithm's results on different kinds of models to different equilibria concepts
- Fine-tune parameters to improve the algorithm's performance on a broad range of models
- Investigate possible variations of the algorithm combining different heuristics of evolutionary optimization algorithms and its result on performance and convergence.

References / Список литературы

1. *M.K.T.B. Paul* Frihauf "Nash Equilibrium Seeking in Noncooperative Games," IEEE TRANSACTIONS ON AUTOMATIC CONTROL, vol. 57, no. 5, pp. 1192-1207, May 2012.
2. *M. Richard, D. McKelvey* "Computation of Equilibria in Finite Games," in Handbook of Computational Economics, Elsevier, 1996, pp. 87-142.
3. *R.W. Srihari Govindan* "Computing Nash equilibria by iterated polymatrix approximation," Journal of Economic Dynamics & Control, pp. 1229-1241, 2004.
4. *R.W. Srihari Govindan* "Computing Nash equilibria by iterated polymatrix approximation," Journal of Economic Dynamics & Control, pp. 1229-1241, 2004.
5. *S.U. Jacek, B. Krawczyk* "Relaxation algorithms to find Nash equilibria with economic applications," Environmental Modeling and Assessment, pp. 63-73, 2000.
6. *M.J. Neely* "A Lyapunov Optimization Approach to Repeated," in Proc. Allerton Conference on Communication, Control, and Computing, 2013
7. *K.H.J. a. D.M.S. Milos S. Stankovic* "Distributed Seeking of Nash Equilibria With," IEEE TRANSACTIONS ON AUTOMATIC CONTROL, pp. 904-919, 2012.
8. *B. Stengel* "Computation of Nash equilibria in finite games: introduction to the symposium," Economic Theory, 2010.

9. *F.B.E.B.K. Mattheos, K. Protopapas* "Coevolutionary Genetic Algorithms for Establishing Nash Equilibrium in Symmetric Cournot Games," Advances in Decision Sciences, 2010.
10. *D.D. Rodica Lung* "An Evolutionary Model for Solving Multiplayer Noncooperative Games," in Knowledge Engineering: Principles And Techniques; Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques, KEPT2007, Cluj-Napoca (Romania), 2007.
11. *T.-L.Y. Wei-Kai Lin* "The Co-Evolvability of Games in Coevolutionary Genetic Algorithms," Taiwan Evolutionary Intelligence Laboratory (TEIL), Taiwan, 2009.
12. *I.A. Ismail* "Game Theory Using Genetic Algorithms," in Proceedings of the World Congress on Engineering 2007, WCE 2007, London, U.K., 2007.
13. *J.C. Olivier Bournez* "Learning Equilibria in Games by Stochastic Distributed Algorithms," in Computer and Information Sciences III, London, UK, Springer-Verlag London, 2013, pp. 31-38.
14. *H.G. Veisi* "A Multi-Modal Coevolutionary Algorithm for Finding All Nash Equilibria," Ubiquitous Information Technologies and Applications, pp. 21-29, 2012.
15. *R.E. Marks* "Playing Games with Genetic Algorithms," Evolutionary Computation in Economics and Finance, pp. 31-44, 2002.
16. *M.A. Suheyla Ozyildirim* "Learning the optimum as a Nash equilibrium," Journal of Economic Dynamics & Control, pp. 483-499, 2000.
17. *R. Axelrod* "The Evolution of Strategies in the Iterated Prisoner's Dilemma," in Genetic Algorithms and Simulated Annealing, Los Altos, CA, Morgan Kaufman, 1987, pp. 32-41.
18. *O.M. John von Neumann* Theory of Games and Economic Behavior, Princeton University Press, 1944.
19. *J. Nash* "Non-Cooperative Games," The Annals of Mathematics, pp. 286-296, 1951.
20. *M. Preuss* Multimodal Optimization by Means of Evolutionary Algorithms, Münster, Germany: Springer International Publishing Switzerland, 2015.
21. *K.A.D. Jong* Evolutionary Computation, Cambridge, Massachusetts; London, England: The MIT Press, 2006.
22. *H. Peters* Game Theory, A Multi-Leveled Approach, Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2015.
23. *L. Bull* "On coevolutionary genetic algorithms," Soft Computing, pp. 201-207, 2001.
24. *L.B.C. Fogarty* "Co-Evolving Communicating Classifier Systems for Tracking," in Artificial Neural Nets and Genetic Algorithms, Vienna, Springer, 1993.
25. *P. Husbands* "Distributed Coevolutionary Genetic Algorithms for Multi-Criteria and Multi-Constraint Optimisation," in Evolutionary Computing. AISB EC 1994. Lecture Notes in Computer Science, vol 865, Berlin, Heidelberg, Springer, 1994.
26. *J.B.P. Sevan, G. Ficici* "A Game-Theoretic and Dynamical-Systems Analysis of Selection Methods in Coevolution," IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, pp. 580-601, 2005.
27. *J.M. Smith* Evolution and the Theory of Games, Cambridge University Press, 1982.
28. *C.B.B. Morsky* "Truncation selection and payoff distributions applied to the replicator equation," Journal of Theoretical Biology, pp. 383-390, 2016.
29. *T. Roughgarden* "Computing equilibria: a computational complexity perspective," Economic Theory, pp. 193-236, January 2010.
30. *V.V. Noam Nisan* Algorithmic Game Theory, New York, USA: Cambridge University Press, 2007.
31. *T. Ken'ichiro Tanakaa* "Discrete approximations of continuous distributions by maximum entropy," Economics Letters, pp. 445-450, 2012.
32. *Y.L. Jian Chi* "Multi-objective Genetic Algorithm based on Game Theory and its Application," in 2nd International Conference on Electronic & Mechanical Engineering and Information Technology (EMEIT-2012), Paris, France, 2012.
33. *J.P.M. Sefrioui* "Nash Genetic Algorithms : examples and applications," in Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), La Jolla, CA, USA, 2000.
34. *.-Y.K. Kwee-Bo Sim* "Solution of multiobjective optimization problems: coevolutionary algorithm based on evolutionary game theory," Artificial Life and Robotics, pp. 174-185, 2004.
35. *E. Cantú-Paz* Efficient and Accurate Parallel Genetic Algorithms, Springer US, 2001.
36. *Thomas* "On evolutionarily stable sets," Journal of Mathematical Biology, pp. 105-115, 1985.
37. *Thomas* "Evolutionary stability: States and strategies," Theoretical Population Biology, pp. 49-67, 1984.
38. *J. Maynard Smith and G. Price* "The logic of animal conflict," Nature, pp. 15-18, 1973.
39. *B. Thomas* "Evolutionarily Stable Sets in Mixed-Strategist Models," Theoretical Population Biology, pp. 332-341, 1985.
40. *R. Selten* "A Note on Evolutionarily Stable Strategies in Asymmetric Animal Conflicts," in Models of Strategic Rationality, Springer Netherlands, 1988, pp. 67-75.
41. *L. Samuelson* Evolutionary Games and Equilibrium Selection, Cambridge, MA: The MIT Press, 1997.

42. T. G. Daniel Kahneman, Heuristics and Biases : The Psychology of Intuitive Judgment, New York, NY, USA: Cambridge University Press, 2002.
43. *Smith* The Wealth of Nations, London, UK: W. Strahan and T. Cadell, 1776.
44. *Murphy* "John Law and Richard Cantillon on the circular flow of income," Journal of the History of Economic Thought, pp. 47-62, 1993.
45. *J.M. Keynes* The General Theory of Employment, Interest and Money, Palgrave Macmillan, 1936.
46. *Marshall* Principles of Economics, London, UK: Macmillan, 1890.
47. *R. Lucas* "Econometric Policy Evaluation: A Critique," in The Phillips Curve and Labor Markets, New York, American Elsevier, 1976, pp. 19-46.
48. *S.A.T. Rizvi* "The Sonnenschein-Mantel-Debreu," History of Political Economy, pp. 228-245, 2006.
49. *R.R. Nelson* "Behavior and cognition of economic actors in evolutionary economics," Journal of Evolutionary Economics, 2015.
50. *A.N. Moshe Hoffman* "An experimental investigation of evolutionary dynamics in the Rock-Paper-Scissors game," Scientific Reports, 2005.
51. *C. Morrison* "Cournot, Bertrand, and modern game theory," Atlantic Economic Journal, pp. 172-174, 1998.
52. *M.M. Iris Lorscheid* "Agent-based mechanism design – investigating bounded rationality concepts in a budgeting context," Team Performance Management, pp. 13-27, 2017.