Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 165 – 172

ISBN 978-83-60810-14-9
ISSN 1896-7094

Evolution of Strategy Selection in Games by Means of Natural Selection

Daniel L. Kovacs, Tadeusz Dobrowiecki
Budapest University of Technology and Economics, H-1117 Budapest, Hungary
Email: {dkovacs, dobrowiecki}@mit.bme.hu

***Abstract*—In this paper we present a novel agent-based simulation model for the natural selection of replicating agents whose survival depends on their programs for selecting their strategy to interact with each other. Game theoretic models describe this kind of interaction. The simulator can be used both for analysis, design and verification of autonomous systems (by intuitively abstracting them into our model and running the simulation). Good system design can be selected even for difficult, underspecified design problems. Although the inspiration of the model comes from evolutionary game theory, the evolving agents may represent not only biological, but also technical systems (e.g. software agents), and thus in some cases the underlying evolutionary mechanisms and observations differ from the typically published results.**

I. Introduction

THIS article presents a novel, realistic agent-based simulation model for the natural selection of replicating agents whose survival depends on their programs for selecting their strategy to interact with each other [1]. Compact game theoretic models describe such interaction, masking most low-level details, but nevertheless catching the essential program features, and simplifying simulation [2].

The implementation of the proposed simulation model is a domain-independent problem-solving tool that facilitates not only analysis, but also design and verification of autonomous systems. Assuming that there is a fixed set of different *design alternatives* (e.g. different types of software agents) in the real environment which interact with each other repeatedly and randomly in a pairwise manner, and that their number depends on their success in these interactions, they can be modeled – including their environment – by **(1st)** a fixed pool of game playing agents (being able to replicate and terminate) and **(2nd)** a 2-person game. These are the two inputs of our simulation model as shown in Fig. 1.

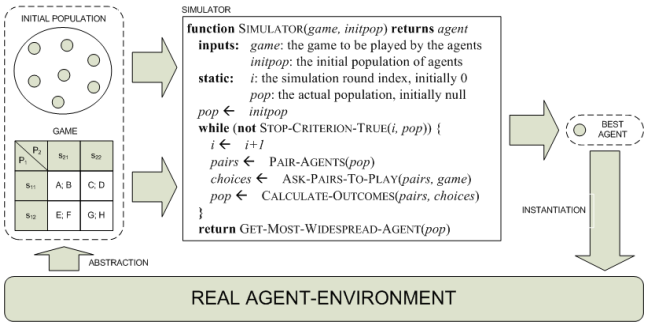

Fig. 1 The simulation model and its use in system analysis and design

The simulation is boldly as follows: agents are repeatedly and randomly paired to play the 2-person game, and their survival (and chance for replication) depends on the utility they can gather during that process. Thus quantitative aspects enter the simulation via the proportion of subpopulations of these different agents constituting the initial population. Evolution essentially means the dynamics of the distribution of these different agents (species) within the evolving population. This way we intend to find the agent corresponding to the best design alternative even when the under-defined specification (usually due to missing, or uncertain knowledge about the task environment) does not make it possible to pinpoint the optimal design via the traditional design process. The success of this process depends very much on the abstracting phase, where the real problem is represented as an input for our simulation model.

The inspiration for the model comes mainly from evolutionary game theory, and the seminal experiments of Robert Axelrod with the **T**it-**F**or-**T**at (**TFT**) strategy in the repeated **P**risoner's **D**ilemma (**PD**) [3]–[5]. Several other models of program evolution exist, but our approach differs from them mainly in the following [6]–[8].

1. We utilize natural selection as a realistic "driving force", and not some preprogrammed, explicit mechanism, to evolve the population;
2. we do not consider the emergence of new variants, only proliferation, i.e. asexual replication (spreading);
3. we do not impose any formal constraints on the structure and inner workings of the agent programs.

We hope thus to model, predict, and support design in several interesting real world situations (e.g. the success of given software agents managing resources or being engaged in electronic commerce on the Internet).

The rest of the paper is organized as follows: Section 2 introduces the background of the simulator, Section 3 describes its architecture and implementation in detail, Section 4 presents and evaluates some essential experiments, Section 5 contains conclusions and outlines future research.

II. Background

In the following, we will briefly summarize those approaches, which mainly influenced our model. The purpose of this is to introduce some fundamental concepts, and to enable later discussion of similarities and differences.

A. Axelrod's experiments

The goal of Axelrod's experiments was to find a program which efficiently plays the repeated PD game (cf. Table I). At first 15, then 62 programs were chosen, and pairwise compared. Every program played against each other a fixed number of rounds. In every round they had to choose between two strategies (cooperate or defect), and got their respective payoff according to their collective choice.

TFT, a simple program, which initially cooperates, and then repeats the previous choice of its opponent, won both tournaments by collecting the most at the end. Axelrod concluded, that because of the importance of PD as a model of social interaction, the core characteristics of cooperation in general must be those found in the TFT.

Axelrod then conducted other experiments to confirm the success of TFT, called ecological and evolutionary analysis. In the former he examined the dynamics of a population composed of programs submitted for the second tournament (all having an equal proportion of the initial population). The proportion of a program changed according to the proportion-weighted average of its scores against other programs in the second tournament. The results of this experiment again underpinned the success of TFT. Its proportion was the largest, and it grew steadily.

In the latter experiment Axelrod used genetic algorithms to evolve programs (represented as bit strings) that could play against the programs submitted for the second tournament [9]. Fitness was equal to the then average payoff. The algorithm produced programs that played effectively against the tournament programs, and resembled TFT.

B. Evolutionary Game Theory

Evolutionary game theory, on contrary to the previous simulations, enables formal analysis and prediction of such evolving systems (although only for relatively simple cases).

For example, let's suppose that we have an infinite population of agents, who strive for resources. The game is divided into rounds, and in every round every agent randomly (according to uniform distribution) meets an other agent to decide upon a resource of value $V>0$. For the sake of simplicity let's say, that there are only two types of agents: hawks (aggressive), and doves (peaceful).

When two hawks meet, they fight, which costs C , and so they get $(V-C)/2$ per head. When two doves meet, they divide the resource equally between each other (they get $V/2$ per head). When a hawk meets a dove, then the hawk takes the resource (gets V), while the dove is plundered (gets 0).

TABLE I.
PAYOFF MATRIX OF THE "PRISONER'S DILEMMA" GAME USED BY AXELROD

Player 2 / Player 1	Defect	Cooperate
Defect	1; 1	5; 0
Cooperate	0; 5	3; 3

This situation is simply modeled by the **H** awk- **D** ove (**HD**) game (cf. Table II) [10]. The gained payoffs are collected over rounds, and the proportion of hawks and doves depends on their average collected payoff. Formally this is as follows.

Let p^i_H and p^i_D denote the proportion, while W^i_H and W^i_D the average collected payoff of hawks and doves in round i respectively. For every $i\geq 0$ $p^i_H, p^i_D \in [0,1]$, and $p^i_H + p^i_D = 1$ is true. The average collected payoff of the whole population in round i , W^i , is

$$W^i = p^i_H W^i_H + p^i_D W^i_D \tag{1}$$

The proportion of hawks and doves in round $i+1$ is calculated according to discrete replicator dynamics.

$$p^{i+1}_H = p^i_H \frac{W^i_H}{W^i} \text{ , and } p^{i+1}_D = p^i_D \frac{W^i_D}{W^i} \tag{2}$$

The average collected payoff of hawks and doves in round $i+1$ is respectively.

$$W^{i+1}_H = W^i_H \left(p^{i+1}_H (V-C)/2 + p^{i+1}_D V \right) \tag{3}$$

$$W^{i+1}_D = W^i_D + p^{i+1}_D V/2 \tag{4}$$

Consequently, for example, it can be simply shown that if $V > C \geq 0$, $W^0_H = W^0_D > 0$, and $p^0_H > 0$, then

$$\lim_{i\to\infty} p^i_H = 1 \text{ , and } \lim_{i\to\infty} p^i_D = 0 \tag{5}$$

This means, that the system converges to a state, where only hawks remain in the population. Many similar, interesting results can be obtained this way, although the necessary assumptions are usually unrealistic, and overly simplified (infinite number of agents; trivial programs, that can be handled analytically; etc) [11]. For more realistic and complex cases, with arbitrary programs (like in the ecological analysis of Axelrod) and finite, overlapping generations of varying size, we need to use simulations.

III. SIMULATION DRIVEN BY NATURAL SELECTION

The proposed simulation model combines the advantages of the previous approaches without their drawbacks. It resembles artificial life in many aspects, but it is different in its purpose (it tries to capture the key features of not only biological, but also technical systems' evolution) [12]. It is an extension to the previous approaches, differing from them in the following.

TABLE II.
PAYOFF MATRIX OF THE "HAWK-DOVE" GAME

Player 2 / Player 1	Hawk	Dove
Hawk	(V-C)/2; (V-C)/2	V; 0
Dove	0; V	V/2; V/2

1. Populations are finite, and vary in size;
2. Generations are overlapping;
3. Agents are modeled individually;
4. The selection mechanism, and the fitness of agents is not explicitly given, but it is a product of agents' features and their interaction;

5. Modeling of not only biological, but also technical systems' evolution is considered.

These differences make the model more realistic, and enable us to use it not only for analysis, but also for design.

C. Detailed Description of the Simulation Model

The basis of the model is an intuitive combination and extension of the ideas discussed in Section 2. The simulation is divided into rounds. There is a finite number of agents in the population, who are randomly paired in every round (according to uniform distribution) to play a 2-person game in the role of one of the players. Every agent of the population plays the same type of game (e.g. just PD, or just HD) in every round of a run, and each of them has a program (e.g. TFT, Random, Always-Cooperate, Always-Defect) for selecting its strategy in these plays.

After a pair of agents finished to play in a given round, the respective (possibly negative) payoffs are added to their individually cumulated utility. If the utility of an agent gets below a level (e.g. zero), then the agent dies, i.e. it instantly disappears from the population, and won't play in the following rounds; otherwise it may reproduce depending on its reproduction strategy. Every agent has a reproduction strategy (defining how, and when to reproduce), but only asexual proliferation, i.e. replication without change is considered. After every agent finished the given round (died, survived, or even replicated), comes the next round.

It can be seen, that in this model there is no explicit selection mechanism directly controlling the proportion of agents in the population (like replicator dynamics, or roulette wheel), but it is an implied, emergent phenomena. Every agent has its own lifecycle (they are born with given features, interact with each other, change, maybe even reproduce, and then possibly die), and only those are "selected" for reproduction, whose features make them able to survive in the environment up to that moment. In our view this process is real *natural selection*.

Of course there are many other more or less similar definitions in literature originating mostly from Darwin [13]. For instance, it is usual to say that "natural selection is a process that affects the frequency distributions of heritable traits of a population" [14] (page 16), and that "heritable traits that increase the fitness of their bearers increase in frequency in a population" [15] (page 821), etc. These statements do not contradict our views, but they aren't specific enough in the sense, that they allow even the selection behind a genetic algorithm to be called "natural". The reason for that is that these statements stem from biology and genetics, which are concerned with natural systems [16], [17]. This is why we use a "new" definition.

Moreover, we need to differentiate between modeling natural selection of natural, and technical systems. In respect of technical systems, "natural selection" may reflect the "natural" peculiarities of technical application areas, which may be quite far from what can be considered natural in biological, or social science. We differentiate these aspects by introducing agents' *reproduction strategies*.

We'll consider two types of reproduction: type 1 is called "natural", and type 2 is called "technical". Agents with type 1 reproduction can have only a limited number of offsprings in their lifetime (maximum one per round). They replicate, if their utility exceeds a given limit (limit of replication). After replication, their utility is decreased with the cost of replication (which is usually equal to the limit of replication). Offsprings start with zero utility, and the same program, and features, as their parents originally (i.e. the same lower limit of utility necessary for survival, limit and cost of replication, and limit on the number of offsprings).

On the other hand, agents with type 2 reproduction can have unlimited offsprings (maximum one per round). They replicate when their utility exceeds the limit of replication, but this limit is doubled every time an offspring is produced, and their utility does not decrease. Offsprings start with the same utility, program, and features, as their parents at the moment of replication (i.e. the same lower limit of utility necessary for survival, and limit of replication).

The rationale behind the above design choices originates in the following: biological agents can reproduce only a limited number of times during their lifetime, while technical systems (e.g. software agents) usually do not have such limitations. The replication (copy) of a technical system is usually inexpensive compared to the replication of a biological agent. Offsprings of technical systems may be exact copies of their parents with the same level of quality, while offsprings of a biological system usually develop from a lower (immature) level. The level of "maturity", where a biological system can reproduce, may be constant, and the same as its parents', but it may increase in the case of a technical system (to represent the increasing cost of production, or the amount of resources needed, etc).

D. Using the Simulation Model for Analysis and Design

As mentioned before, the simulation model can be used not only for description, but also for design of technical systems. Let's suppose, for example, that we want to design an efficient software agent (e.g. a broker agent, or a trading agent on the Internet acting on behalf of human users, trying to maximize their profit). Our goal is to make this agent as efficient and useful as possible in hope that in result it will spread better (e.g. more users will start to use it). Of course there are many other factors (like marketing, ergonomics, etc) that may influence the success of such a system, but, as for a Designer, ensuring functional efficiency and usefulness is a primary objective. Nonetheless these properties depend not only on the program of the agent, but also on its environment (e.g. the other concurrent agents), which may be not fully accessible, too complex and dynamic to be modeled and analyzed exactly in order to calculate an optimal system design (e.g. in the case of Internet).

It is inevitable to abstract the problem. We recommend game theory to model the strategic interaction in the real agent-environment. It may mask most of the subtlety of agents' programs, but it catches essential features, and works toward a simpler simulation, which becomes necessary, if – according to the conclusions of Section 2 – agents' programs are too complex to be handled analytically.

For example, let's suppose that software agents participate in electronic auctions on the Internet to acquire goods for

their human users. The less they pay for them, the higher their individual utility is (from the perspective of their users). If two agents agree to cooperate (e.g. to raise the bid only until one of them wins and then share equally), then they pay less, than being non-cooperative (e.g. when competing against each other by bidding more and more to win). The best outcome is however for an agent to "cheat" its cooperative partner by defecting (e.g. raising the bid to win while the other is not competing and then not sharing the win). This outcome is the worst for the other agent since it gets nothing. Let's say, that if a software agent is successful, then after it achieves a given profit for its user, then its user invests in another such agent, but if the profit is below a level then she/he stops using it. *What software agent would we design to better survive and proliferate in this scenario?*

The situation can be modeled by the proposed simulation: agents have two strategies (cooperate or defect), and the order of their preferences corresponds to a PD game (cf. Section 4/A). Now we can estimate the distribution of the various software agents in the real agent-environment, and construct an initial population accordingly. Assuming that natural selection is the "driving force" behind agents' evolution, we can give the corresponding PD game and the initial population as an input to the simulation model, and run it. If we run the simulation by injecting several different candidate solutions into that initial population, we can get the most successful variant. This way we can use natural selection to support not only analysis, but also design.

E. Some Thoughts About Natural Selection

But why do we need natural selection? Why don't we use some other, explicit model (e.g. replicator dynamics) for simulating the dynamics of the real environment? – First, we claim that using natural selection instead of some explicitly declared mechanism for evolving the population of agents allows more realistic predictions of several interesting real world phenomena. In Section 4 we'll discuss this in more detail (in relation to experiments concerning replicator dynamics). Second, if the selection mechanism is direct and explicit, then it usually accounts for some kind of fitness to measure the goodness of individuals. Such a value would reflect the goodness of all (maybe even hidden or implicit) features influencing the success of an agent in its environment. The calculation of such a fitness value can be hopeless for complex, dynamic systems and environments.

In the proposed simulation we currently use a cumulated utility value associated with the goodness of an agent. But it is not a fitness value. The fitness of an unvarying individual can change only when its environment changes. But this is not true for the cumulated utility, which is only a parameter of the agent's state, and may change even when the agent, its program, or its environment does not vary. Also, the survival of an agent may depend on much more sophisticated terms than just checking if the cumulated utility value is below a lower limit. This mechanism could be easily exchanged by a much more complex and realistic variant.

F. Implementation Issues

The proposed simulation model was implemented in the **JADE** (**J**ava **A**gent **DE**velopment) framework [18]. It is an open-source, Java-based, platform independent, distributed middle-ware and **API** (**A**pplication **P**rogramming **I**nterface) complying with the **FIPA** (**F**oundation for **I**ntelligent **P**hysical **A**gents) standards [19]. It enables relatively fast and easy implementation of physically distributed, asynchronous, high level multi-agent systems.

The implemented software architecture was aimed to be as simple, as possible. It consisted of only two JADE agents: a `GameAgent` (`GAg`), and a `PlayerAgent` (`PAg`). `GAg` was responsible for conducting the runs of the simulation, and orchestrating `PAg`-s, while `PAg`-s were the actual agents in the population, who were paired in each round to play a given 2-person game.

Each JADE agent had a variety of (mostly optional) startup parameters, which in case of a `GAg` set the type of the game to be played (e.g. PD, or HD, or else), the maximal number of agents in the population, and the maximal number of rounds in the run. The OR-relation of the latter two defined the termination criteria of a run. The startup parameters of a `PAg` set agents' program and reproduction strategy, initial utility, the lower limit of utility, the limit and cost of reproduction, the limit on the number of offsprings, and the capacity of memory. The latter was needed because each agent had to be able to use its percept history in order to decide upon the strategy to be played in a given round. The percept history of an agent associated a series of events (information about past plays) to agent identifiers (ID-s). There was a limit on maximal length of these series and maximal number of ID-s. If any of these limits was exceeded then the oldest element was replaced by the new one.

Now the simulation went as follows. First a given number of `PAg`-s were started on the JADE agent platform (constituting the initial population), followed by a `GAg`, who at the beginning of every round first searched the platform for available `PAg`-s (because later there may have been newly born agents, or some agents terminated), then made a pairing of the `PAg`-s, and informed these pairs about the game to be played (who plays with whom and in what role). The pairs of `PAg`-s then replied to the `GAg` with a chosen strategy ID respectively. The `GAg` then calculated their respective payoff accordingly and informed them about it. This was repeated until the termination criterion was satisfied. Several interesting experiments were conducted this way. Some of them are explained in the next section.

IV. Experiments

The experiments consisted of running the simulation described above with several different initial populations and games to observe the changes in the number, proportion, and average utility of the different types of agent programs. Since JADE is a distributed framework (and for faster performance) a cluster of 10-13 PC-s (with 1-2 Ghz CPU, 0,5-1 Gb RAM, Win. XP, and JADE 3.5) was used to run the simulations.

Each experiment had its own settings, but a part of them was the same for every experiment. The maximal number of agents was 800; the maximal number of rounds was 250; the maximal number of offsprings was 3; the limit and the cost

of reproduction was 20; the lower limit of agents' utility and their initial utility was 0; the maximal number of percept histories (about different opponents) was 1000; and the limit on the length of such a percept history was 4 for every agent in every experiment. Everyone was playing in every round (except when the number of agents was odd).

In the following we will describe these experiments grouped according to the games the agents' were playing. Five games were examined: **P**risoner's **D**ilemma (**PD**), **C**hicken **G**ame (**CG**), **B**attle of **S**exes (**BS**), **L**eader **G**ame (**LG**), and **M**atching **P**ennies (**MP**). The first 4 games are the only interesting (i.e. dilemmas, where the choice is not trivial) between the 12 symmetric out of altogether 78 ordinally different 2×2 games [20]. MP is an addition to these because unlike them it has no pure strategy **N**ash **E**quilibrium (**NE**), and this has some interesting implications (cf. Section 4/E) [21].

During the experiments agent programs were drawn from a fixed set. Only the following programs were studied yet: Always-Strategy1, Always-Strategy2, TFT, and Random. Nonetheless both types of agents' reproduction strategy (cf. Section 3/A) were examined. All in all, this configuration was more than enough to run insightful experiments comparable to the previous results discussed in Section 2.

G. Prisoner's Dilemma

PD is one of the most popular 2-person games in game theory [22]. It is a special case of the HD game, when $V > C \geq 0$ (cf. Table II). The original story of the game is about two prisoners, who are put in separate cells (cannot communicate), and are asked to simultaneously decide, whether to cooperate, or defect. The best outcome is defecting, if the other cooperates, but it is the worst outcome for the other. It is better if both defect, and even better, if both cooperate. The game is called a "dilemma" because its only NE is the sub-optimal Defect-Defect outcome.

Similar situations may arise in the world of artificial systems too (cf. example in Section 3/B). If the payoffs are chosen according to the HD game, where V=4, C=2 (and so it becomes a PD), then if the initial population consists of altogether 6 agents: 3 Always-Cooperate, and 3 Always-Defect agents, then the proportions of the different agent programs change according to Fig. 2, which is in accordance with the formal predictions of Section 2/B (cf. Eq. 5). Defective agents (hawks) infest the population, and the proportion of cooperative players (doves) steadily converges to zero. The reproduction strategy of agents in Fig. 2 is of type 1 ("natural"), but essentially the results are the same in case of type 2 ("technical") reproduction.

Fig. 3-5 show the change of the quantity, and average utility, if the initial population consists of 3 Random, and 3 TFT agents. The quantity of these subpopulations ("species") does not decrease because there are no negative payoffs in the game, and so agents cannot die since their cumulated utility cannot decrease below the lower limit.

Fig. 5 shows the same situation as Fig. 4 except that the reproduction is "technical". A periodical change of the average utility of subpopulations can be observed on Fig. 4, while it is monotonously increasing on Fig. 5. That is because "natural" reproduction has a cost (equal to the limit of reproduction) on the contrary to "technical" reproduction.

According to Fig. 3, Random agents typically outperform TFT agents by far. This is the case with both types of reproduction. Similarly, Always-Defect agents also outperform TFT agents. These observations seem to differ from the results mentioned in Section 2/A, although the latter is not so surprising, since the numerical values of the game are now such, that the defective player gets as much against a cooperating player, as two cooperating players together, and its average income (5/2) is better than by cooperating (2/2).

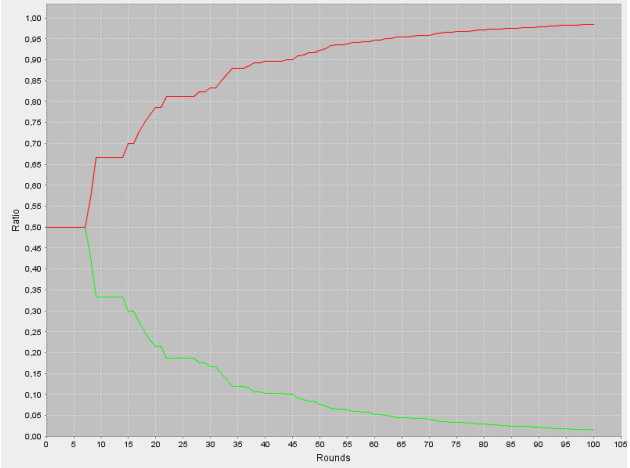

Fig. 2 Typical change of agent programs' proportion in PD, if the initial population consists of 3 Always-Cooperate (green), and 3 Always-Defect (red) agents whose reproduction strategy is "natural".

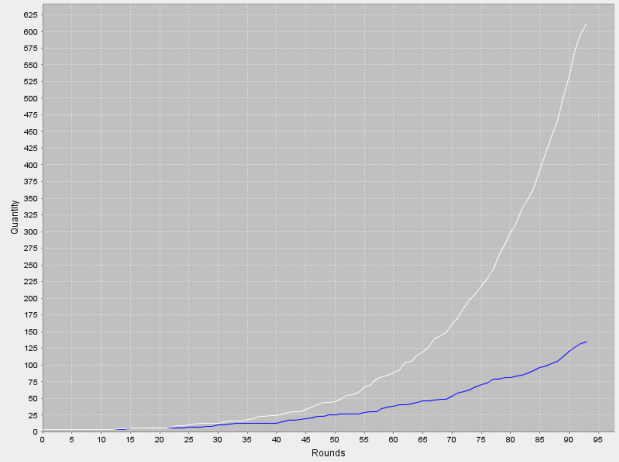

Fig. 3 Typical change of agent programs' quantity in PD, if the initial population consists of 3 Random (white), and 3 TFT (blue) agents whose reproduction strategy is "natural".

Moreover, according to Fig. 3-5, the change of subpopulations' proportion isn't in direct proportionality with the ratio of their average utility and the average utility of the whole population, as predicted by replicator dynamics (cf. Eq. 2). It must be pinned down, that replicator dynamics holds a central role among models of evolutionary dynamics. Some consider it even to be a fundamental theorem of natural selection [23]-[25]. Reference [26] even writes "The dynamics of such replicator systems are described by the fundamental theorem of natural selection". Nonetheless, according to

these experiments, replicator dynamics seems to be unable to model such scenarios properly. Thus, in our view, the assumptions behind Eq. 2 are unrealistic.

If, for example, there are two subpopulations, and one of them grows exponentially (which is common in evolution), then after a while their average utility will determine the average utility of the whole population, and so the ratio of these two averages will converge to 1. In this case replicator dynamics would predict that the growth of the large subpopulation stops (cf. Eq. 2). But why should it?

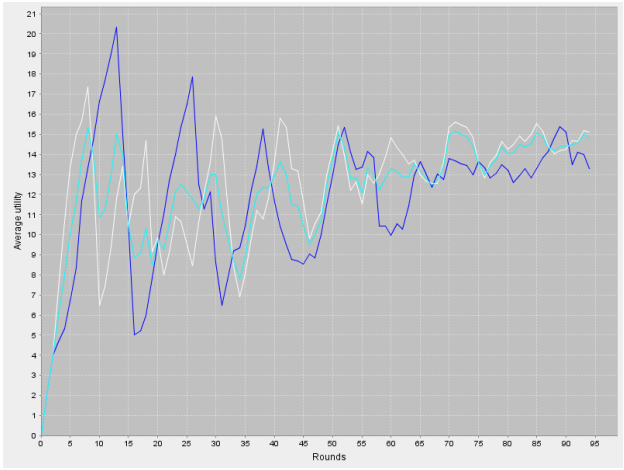

Fig. 4 Typical change of agent programs' average utility in PD, if the initial population consists of 3 Random (white), and 3 TFT (blue) agents whose reproduction strategy is "natural". The average utility of the whole population is also shown (light blue).

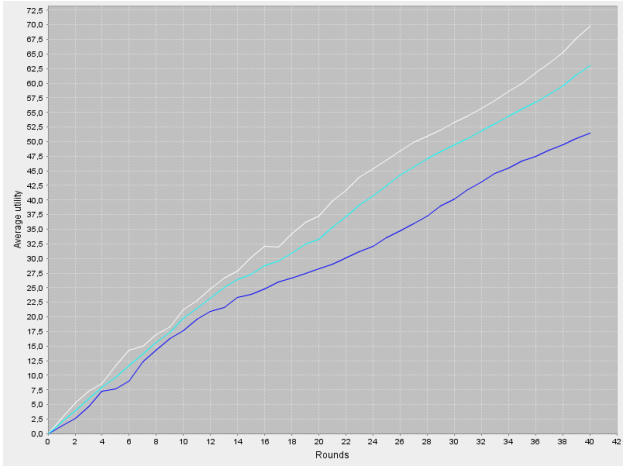

Fig. 5 Typical change of agent programs' average utility in PD, if the initial population consists of 3 Random (white), and 3 TFT (blue) agents whose reproduction strategy is "technical". The average utility of the whole population is also shown (light blue).

The larger the population, the more offsprings are born. The tendency of growth accelerates, if it is not limited by finite resources, or some other way. So maybe there is an implicit assumption (e.g. about finite resources) in Eq. 2. Apparently after a while every evolutionary system should reach its boundaries, but until then their growth must be governed by a dynamics different from replicator dynamics.

The results shown in Fig. 6 utter these concerns further. Axelrod's ecological analysis (cf. Section 2/A), based on replicator dynamics, predicts the extinction of the invading defectors, but our experiments show the opposite is true [7].

All of the above results are typical in a sense that they are indifferent to parameter changes (e.g. changing the size of the initial population; changing the type, the limits, or the cost of reproduction, or the payoff values of the game). TFT agents could be made a little better by increasing their memory, but in the end it doesn't change the overall tendency. These observations may also help to explain the scarce evidence of TFT-like cooperation in nature [27].

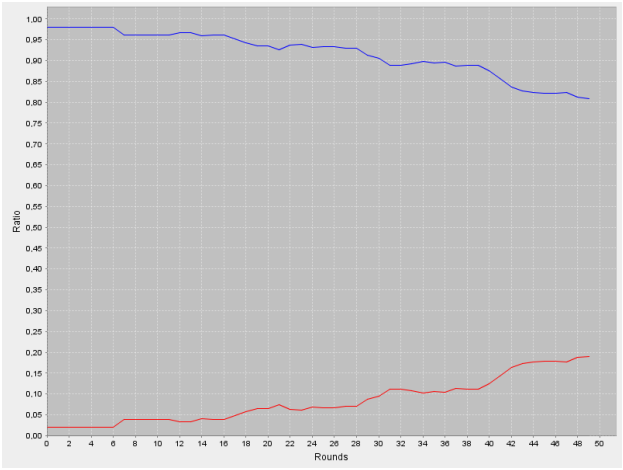

Fig. 6 Typical change of agent programs' proportion in PD, if the initial population consists of 50 TFT (blue), and 1 invading Always-Defect (red) agent(s) whose reproduction strategy is "natural".

H.Chicken game

This game is also a special case of the HD game, when $C > V$ (cf. Table II) [22]. The original story is about two cars driving toward each other. If both drivers are "reckless", and won't swerve away, it is the worst outcome, since they crash. Better is, if one swerves away (being "chicken"), while the other wins (best outcome). But it is better for the former, if the latter swerves away too. This is a mixed motive game, since it has two NE-s (if players do the opposite).

A technical scenario may be the same as in the previous example (cf. Section IV/A) with the exception that if both agents are non-cooperative (reckless), then it is the worst outcome for them (e.g. because if two such cheating bidder agents meet, they disclose each other, and thus are excluded from the auction, and must pay a penalty to the organizers).

Experiments with this game showed different results than in case of PD in Section 4/A. The main reason for that is that the payoffs were chosen according to HD game, where V=2, C=4, and so agents could die because of negative payoffs.

Always-Strategy2 (i.e. Always-Chicken) proved to be the best (most proliferating) program out the four studied alternatives if there were only two types of programs in the initial population. Always-Reckless and TFT claimed the second place, while Random was the worst. This means that if the cost of being mutually defective is beyond the achievable value, then it becomes too risky not to cooperate.

It was interesting to observe, that if Always-Reckless proved to be the winner of a situation (i.e. if it extinguished an other "species"), then it too died out. In this aspect Al-

ways-Reckless is “parasite”, that exploits the other subpopulations from whom its survival depends.

Experiments with more than two types of agent programs were rather unpredictable. They depended mostly on the actual pairing of the individuals in the first dozen of rounds.

I. Battle of Sexes

This game is also a mixed motive game, like CG, but it differs from it in that it is asymmetric by default (cf. Table III) [2]. The original story is about a husband and wife, who must choose between going to a football match, or an opera. The husband would better like to go to the football match, while his wife would better go the opera. But, in any case, it is more preferable for them to be together, than to be alone.

Cooperation is different in the case of the husband, than in the case of the wife. They cooperate, if they try to do what is best for the other, and if they both do that, it is the worst (husband goes to opera, and wife goes to football).

TABLE III.
PAYOFF MATRIX OF THE “BATTLE OF SEXES” GAME

Wife / Husband	Opera	Football
Opera	1, 2	-1, -1
Football	0, 0	2, 1

This means, that Always-Cooperate, and Always-Defect strategies are a bit more complex now, since they depend on the role of the agents too. TFT needs also to be revised.

A corresponding technical scenario could be, when two tasks share the same resource (e.g. CPU, or server). Their user wants them to finish as soon as possible. The worst case is, if they don’t access the resource at all (e.g. its availability may have a default cost to the user). If they access it at the same time (both defect), it is better, but it slows their execution too much. So it is better for them to access the resource separately, but the first to access it is the best.

Experiments showed that regardless of the type of reproduction, Always-Cooperate and TFT agents were the worst (others made them die out every time). Always-Defect was the best program, and Random was second (it survived).

J. Leader game

This game is similar to the symmetrical form of BS, with the exception that mutual defection is the worst outcome, and mutual cooperation is better [28].

The name of the game comes from the following situation: two cars wait to enter a one-way street. The worst case is, if they go simultaneously, because they crash. If both wait (cooperate), it is better. But it is even better if they go separately. The one, who goes first, is the best (cf. Table IV) .

A technical scenario could be similar to the previous example in Section IV/C with the exception that if both tasks access the resource at the same time, then it breaks (e.g. the server crashes). So this is the worst outcome. It is better, if they do not access it at all, and even better, if they access it separately. The first agent to access the resource is the best.

According to our experiments, TFT and Always-Cooperate were better, than Always-Defect agents, but Random agents again outperformed TFT agents. The reproduction strategy made a difference in the tendencies, but not in the outcome.

TABLE IV.
PAYOFF MATRIX OF THE “LEADER” GAME

Player 2 / Player 1	Go	Wait
Go	-1, -1	2, 1
Wait	1, 2	0, 0

K. Matching Pennies

This game is asymmetric (like BS), with the exception that it has no symmetric form, and cooperation and defection have no meaning in it [29]. Thus the first (hitherto cooperative) choice of TFT doesn’t particularly matter now.

The original game is about two players, who both have a penny. They turn the penny secretly to heads or tails, and then reveal their choice simultaneously. If the pennies match, one player gets a dollar from the other, else it is conversely (cf. Table V).

TABLE V.
PAYOFF MATRIX OF THE “MATCHING PENNIES” GAME

Player 2 / Player 1	Heads	Tails
Heads	1, -1	-1, 1
Tails	-1, 1	1, -1

A corresponding technical scenario can be where two software agents compete for resources (e.g. locks to files), and they have two different strategies to get the resources. One of the agents is faster than the other, so if both choose the same strategy, then the faster agent gets all of the resources. Else the slower agent can also get some of them.

Our experiments showed that in this scenario Random agents were the fittest for survival (playing the only mixed NE of the game), but in case of type 1 reproduction they died out like all the others. However in case of type 2 (technical) reproduction they could cumulate enough utility to ensure their survival, and start proliferating after a while.

V. CONCLUSIONS

In this article we presented a novel agent-based simulation model for natural selection of programs selecting strategies in 2-person games, and its use in system analysis and design. Our goal was (1) to create a framework enabling agent design for complex, underspecified environments (e.g. Internet); (2) to give a realistic model for the natural selection of not only natural, but also technical systems; and (3) to reproduce some decisive experiments.

The framework facilitates agent design by supporting the choice of agents (from a finite set of alternatives). The distinction between natural and technical systems is made by introducing reproduction strategies of the agents. Natural selection is not explicitly present, but emerges from the inner workings of the agent population. Experiments revealed several interesting aspects of such population dynamics. For example, the assumptions behind replicator dynamics were shown to be inherently unrealistic and simplifying; the fitness of the Tit-for-Tat strategy was the opposite of what

was expected according to the previous experiments of Axelrod; not just the Prisoner's Dilemma, but several other fundamental 2-person games were examined, and for every such game a concrete technical analogy was also given.

There are many possible ways to continue this research. For instance the simulation can be further refined by allowing agents to play not only one, but several different games during a run. We currently use only 2-person games for modeling strategic interaction between agents. This can be extended to N-person games. More complex programs could be examined in the experiments. We could introduce sexual reproduction instead of the current asexual replication. A genetic representation of agent programs could be given to enable the birth of new variations. Finally, the limits on the cardinality of the agent population could implicitly depend on environmental resource limitations.

References

[1] J. Haldane, "The theory of natural selection today". *Nature,* 183 (4663), pp. 710–713, 1959.

[2] D. Fudenberg and J. Tirole, *Game theory*, MIT Press, 1991.

[3] J. Maynard-Smith, *Evolution and the Theory of Games*, Cambridge University Press, 1982.

[4] R. Axelrod, *The Evolution of Cooperation*, Basic Books, 1984.

[5] R. Axelrod, *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*, Princeton University Press, 1997.

[6] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* . MIT Press, 1992.

[7] C. Ferreira, "Gene Expression Programming: A New Adaptive Algorithm for Solving Problems," *Complex Systems* , vol. 13, no. 2, pp. 87-129, 2001.

[8] D. L. Kovacs, "Evolution of Intelligent Agents: A new approach to automatic plan design", In: Proceedings of IFAC Workshop on Control Applications of Optimization, Elsevier, 2003.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, 1975.

[10] J. Maynard Smith, "Theory of games and the evolution of animal contests," *Journal of Theoretical Biology* , vol. 47, pp. 209-221, 1974.

[11] J. Hofbauer and K. Sigmund, "Evolutionary game dynamics," *Bulletin of the American Mathematical Society*, vol. 40, pp. 479–519, 2003.

[12] M. Bedau, "Artificial life: organization, adaptation and complexity from the bottom up," *TRENDS in Cognitive Sciences*, vol. 7, no.11, 2003.

[13] C. Darwin, *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life* , John Murray , 1859.

[14] J. A. Endler, *Natural Selection in the Wild,* Monographs in Population Biology 21, Princeton University Press, 1986.

[15] P. Kitcher, "Philosophy of Biology," In: *The Oxford Handbook of Contemporary Philosophy* , ed. Frank Jackson and Michael Smith, Oxford University Press, pp. 819-847, 2008.

[16] D. J. Futuyma, *Evolutionary Biology*, Sinauer Associates Inc., 1998.

[17] D. J. Futuyma , *Evolution* . Sinauer Associates Inc., 2005.

[18] F. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*, Wiley Series in Agent Technology, 2007.

[19] Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org

[20] A. Rapoport, M. Guyer, and D. Gordon *The 2X2 Game* , University of Michigan Press, 1976.

[21] J. F. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951.

[22] W. Poundstone, *Prisoner's Dilemma: John Von Neumann, Game Theory, and the Puzzle of the Bomb*, Anchor Books, 1992.

[23] R. A. Fisher, *The Genetical Theory of Natural Selection*, Clarendon Press, Oxford, 1930.

[24] G. R. Price, "Fisher's "fundamental theorem" made clear," *Annals of Human Genetics,* vol. 36, pp. 129-140, 1972.

[25] A. W. F. Edwards, "The fundamental theorem of natural selection". *Biological Reviews,* vol. 69, pp. 443–474, 1994.

[26] J. Neumann, G. Lohmann, J. Derrfuss, and D.Y. von Cramon, "The meta-analysis of functional imaging data using replicator dynamics," *Human Brain Mapping*, vol. 25, no. 1, pp. 165-173, 2005.

[27] P. Hammerstein, "Why is reciprocity so rare in social animals? A protestant appeal" In: P. Hammerstein, Editor, *Genetic and Cultural Evolution of Cooperation*, MIT Press. pp. 83–94, 2003.

[28] M. J. Guyer and A. Rapoport, "Information effects in two mixed motive games," *Behavioral Science*, vol. 14, pp. 467-482, 1969.

[29] D. Fudenberg and D. K. Levine, "Consistency and cautious fictitious play," *Journal of Economic Dynamics and Control*, vol. 19, no. 5-7, pp. 1065-1089, 1995.