A Mathematical Analysis of the Long-run Behavior of Genetic Algorithms for Social Modeling

Ludo Waltman and Nees Jan van Eck

ERIM REPORT SERIES *RESEARCH IN MANAGEMENT*	
ERIM Report Series reference number	ERS-2009-011-LIS
Publication	March 2009
Number of pages	43
Persistent paper URL	http://hdl.handle.net/1765/15181
Email address corresponding author	lwaltman@ese.eur.nl
Address	Erasmus Research Institute of Management (ERIM) RSM Erasmus University / Erasmus School of Economics Erasmus Universiteit Rotterdam P.O.Box 1738 3000 DR Rotterdam, The Netherlands Phone: + 31 10 408 1182 Fax: + 31 10 408 9640 Email: info@erim.eur.nl Internet: www.erim.eur.nl

Bibliographic data and classifications of all the ERIM reports are also available on the ERIM website: www.erim.eur.nl

ERASMUS RESEARCH INSTITUTE OF MANAGEMENT

REPORT SERIES

RESEARCH IN MANAGEMENT

ABSTRACT AND KEYWORDS	
Abstract	We present a mathematical analysis of the long-run behavior of genetic algorithms that are used for modeling social phenomena. The analysis relies on commonly used mathematical techniques in evolutionary game theory. Assuming a positive but infinitely small mutation rate, we derive results that can be used to calculate the exact long-run behavior of a genetic algorithm. Using these results, the need to rely on computer simulations can be avoided. We also show that if the mutation rate is infinitely small the crossover rate has no effect on the long-run behavior of a genetic algorithm. To demonstrate the usefulness of our mathematical analysis, we replicate a well-known study by Axelrod in which a genetic algorithm is used to model the evolution of strategies in iterated prisoner's dilemmas. The theoretically predicted long-run behavior of the genetic algorithm turns out to be in perfect agreement with the long-run behavior observed in computer simulations. Also, in line with our theoretically informed expectations, computer simulations indicate that the crossover rate has virtually no long-run effect. Some general new insights into the behavior of genetic algorithms in the prisoner's dilemma context are provided as well.
Free Keywords	genetic algorithm, long-run behavior, social modeling, economics, evolutionary game theory
Availability	
Classifications	

A mathematical analysis of the long-run behavior of genetic algorithms for social modeling

Ludo Waltman Nees Jan van Eck

Econometric Institute, Erasmus School of Economics

Erasmus University Rotterdam

P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

E-mail: {lwaltman,nvaneck}@ese.eur.nl

Abstract

We present a mathematical analysis of the long-run behavior of genetic algorithms that are used for modeling social phenomena. The analysis relies on commonly used mathematical techniques in evolutionary game theory. Assuming a positive but infinitely small mutation rate, we derive results that can be used to calculate the exact long-run behavior of a genetic algorithm. Using these results, the need to rely on computer simulations can be avoided. We also show that if the mutation rate is infinitely small the crossover rate has no effect on the long-run behavior of a genetic algorithm. To demonstrate the usefulness of our mathematical analysis, we replicate a well-known study by Axelrod in which a genetic algorithm is used to model the evolution of strategies in iterated prisoner's dilemmas. The theoretically predicted long-run behavior of the genetic algorithm turns out to be in perfect agreement with the long-run behavior observed in computer simulations. Also, in line with our theoretically informed expectations, computer simulations indicate that the crossover rate has virtually no long-run effect. Some general new insights into the behavior of genetic algorithms in the prisoner's dilemma context are provided as well.

Keywords

Genetic algorithm, long-run behavior, social modeling, economics, evolutionary game theory.

1 Introduction

The field of evolutionary computation is concerned with the study of all kinds of evolutionary algorithms. These algorithms can be used for various purposes. Perhaps the most popular purpose for which they can be used is function optimization (e.g., [24, 27, 36]). In the function optimization context, evolutionary algorithms can be seen as heuristics that serve as alternatives to more traditional techniques from the fields of combinatorial optimization and mathematical programming. Another important purpose for which evolutionary algorithms can be used is the modeling of biological and social phenomena (e.g., [39]). This is the topic with which we are concerned in this paper. Our focus is in particular on the use of evolutionary algorithms for modeling social phenomena.

When using evolutionary algorithms in the social modeling context, one of the assumptions one makes is that the agents whose behavior is being modeled are boundedly rational. This basically means that the agents are assumed not to behave in a utility maximizing manner. There are numerous ways in which boundedly rational behavior can be modeled (e.g., [15, 23]). A popular approach is to rely on an evolutionary metaphor. This is the approach that is taken by evolutionary algorithms. In its simplest form, the evolutionary approach assumes that there is a population of agents and that for each agent in the population the strategy it uses depends on the population-wide past performance of strategies. The better the past performance of a strategy, the more likely the strategy is to be used again. The evolutionary approach also assumes that there always is a small probability that an agent experiments with a new strategy.

The evolutionary approach to modeling boundedly rational behavior has attracted a lot of attention, not only from researchers in the field of evolutionary computation but also from researchers in the social sciences, in particular from economists. Traditionally, economists have typically relied on game-theoretic models to analyze interactions between agents. These models assume agents to behave in a fully rational way. Nowadays, however, the limitations of game-theoretic models are well recognized and many economists have started to study evolutionary models of agent behavior. These models are based on the assumption that the behavior of agents can best be described using some evolutionary mechanism rather than using the idea of full rationality.

In the field of economics, there are two quite separate streams of research that are both concerned with the evolutionary approach to modeling boundedly rational behavior. One stream

of research, which is usually referred to as agent-based computational economics (e.g., [46]), makes use of techniques from the field of evolutionary computation. Especially genetic algorithms (GAs) are frequently used. Early work in this stream of research includes [5–7, 19, 29, 34, 37, 38], and examples of more recent work are [1, 3, 25, 28, 33, 54, 55]. The other stream of research is more closely related to traditional game theory and is referred to as evolutionary game theory (e.g., [26, 35, 51, 56]). Like the traditional game-theoretic approach, the evolutionary game-theoretic approach is model-based and relies heavily on mathematical analysis. The use of computer simulations is not very common in evolutionary game theory.

In this paper, it is not our aim to argue in favor of either the agent-based computational economics approach, which emphasizes algorithms and computer simulations, or the evolutionary game-theoretic approach, which emphasizes models and mathematical analysis. Instead, we want to show how the former approach can benefit from the mathematical techniques used in the latter approach. More specifically, we want to show how evolutionary algorithms that are used for modeling social phenomena can be analyzed mathematically using techniques that are popular in evolutionary game theory. Our focus in this paper is on one particular type of evolutionary algorithm, namely GAs with a binary encoding. However, we emphasize that the approach that we take can be applied to other types of evolutionary algorithms as well. The reason for focusing on GAs with a binary encoding is that this seems to be the type of evolutionary algorithm that is used most frequently for modeling social phenomena (e.g., [1–7, 10, 12, 18, 19, 25, 30, 33, 34, 37, 38, 50, 54, 55, 57]).

The mathematical analysis that we present in this paper deals with the long-run behavior of GAs with a binary encoding. The GAs are assumed to be used in the social modeling context (for theoretical work on GAs in the function optimization context, see e.g. [39, 42–44, 53]). In the terminology of [54], we are concerned with GAs that are used for modeling social learning (as opposed to individual learning). Our work can be seen as an extension of the work of Dawid [19], who derived a number of important mathematical results on the behavior of GAs. For small and moderate population sizes, the results of Dawid do not provide a full characterization of the long-run behavior of GAs. We extend the work of Dawid by deriving results that do provide a full characterization of the long-run behavior of GAs for small and moderate population sizes. Using our results, the long-run behavior of a GA can be calculated exactly and needs not be estimated using computer simulations. This means that it is no longer necessary to run a GA a

large number of times for a large number of iterations in order to get insight into its long-run behavior. The use of our mathematical results has at least three advantages over the use of computer simulations:

(1) Our mathematical results can be used to calculate the long-run behavior of a GA exactly, while computer simulations can only be used to estimate the long-run behavior of a GA.

(2) When using computer simulations, it can be difficult to determine how many iterations of a GA are required to approximate the long-run behavior of the GA reasonably closely. Our mathematical results do not have this problem.

(3) Calculating the exact long-run behavior of a GA using our mathematical results requires less computing time than obtaining a reasonably accurate estimate of the long-run behavior of a GA using computer simulations.

Our mathematical results have one important limitation, which is that on most of today's computers they can only be used if the chromosome length is not greater than about 24 bits. If the chromosome length is greater than about 24 bits, the use of our mathematical results to calculate the long-run behavior of a GA most likely requires a prohibitive amount of computer memory.

Like in [19], the mathematical analysis presented in this paper relies on the assumption that the mutation rate is positive but infinitely small. (In other words, the analysis is concerned with the limit case in which the mutation rate approaches zero.) In simulation studies with GAs, researchers typically work with values between 0.001 and 0.01 for the mutation rate. This seems to be a rather pragmatic choice (cf. [19]). On the one hand, lower values for the mutation rate would lead to very slow convergence and, consequently, very long simulation runs. On the other hand, higher values for the mutation rate would lead to convergence to unstable, difficult to interpret outcomes. We believe that our assumption of an infinitely small mutation rate is justified because an infinitely small mutation rate is less arbitrary than a mutation rate whose value is determined solely based on pragmatic grounds (cf. [21]). The assumption of an infinitely small mutation rate is also in line with the common practice in evolutionary game theory, in which a similar assumption is almost always made. The advantage of assuming an infinitely small mutation rate is that it greatly simplifies the mathematical analysis of the long-run behavior of GAs (see also [19]). In fact, GAs with an infinitely small mutation rate can be analyzed in a similar way as well-known models in evolutionary game theory (e.g., [21, 31, 52, 58]). Like in

evolutionary game theory, mathematical results provided by Freidlin and Wentzell [22] are the key tool for analyzing the long-run behavior to which convergence will take place. We note that, in addition to the assumption of an infinitely small mutation rate, there are some other assumptions on which our mathematical analysis relies. However, these assumptions are quite mild. Most GAs will probably satisfy them, and if a GA does not satisfy them, a minor modification of the GA will usually be sufficient to meet the assumptions.

To demonstrate the usefulness of our mathematical analysis, we replicate a well-known study by Axelrod [12] (reprinted in [13]; see also [19, 39]). Axelrod used a GA to model the evolution of strategies in iterated prisoner's dilemmas. He showed that an evolutionary mechanism can lead to cooperative behavior. Axelrod's study has been one of the first and also one of the most influential studies on the use of evolutionary algorithms for modeling social phenomena. Directly or indirectly, his study seems to have inspired many researchers (e.g., [4, 8–10, 16–18, 20, 30, 40, 41, 47, 50, 57]). The results obtained by Axelrod are all based on computer simulations. In this paper, we show that more or less the same results can be calculated exactly, with no need to rely on simulations. We also discuss some new insights that exact calculations provide.

The mathematical analysis that we present in this paper also has an important implication for the choice of the parameters of a GA. The analysis indicates that if the mutation rate is infinitely small the crossover rate has no effect on the long-run behavior of a GA. This is a quite remarkable result that, to the best of our knowledge, has not been reported before in the theoretical literature on GAs. The result implies that when GAs are used for modeling social phenomena the crossover rate is likely to be a rather insignificant parameter, at least when one is mainly interested in the behavior of GAs in the long run (for the short run, see [47]). This suggests that in many cases the crossover rate can simply be set to zero, in which case no crossover will take place at all. Simulation results that we report in this paper indeed show no significant effect of the crossover rate on the long-run behavior of a GA.

The remainder of this paper is organized as follows. In Section 2, we present a mathematical analysis of the long-run behavior of GAs that are used for modeling social phenomena. Based on the analysis, we derive an algorithm for calculating the long-run behavior of GAs in Section 3. In Section 4, we demonstrate an application of the algorithm by replicating Axelrod's study [12]. Finally, we discuss the conclusions of our research in Section 5. Proofs of our

mathematical results are provided in the appendix.

2 Analysis

The general form of the GAs that we analyze in this paper is shown in Figure 1. In this figure, and also in the rest of this paper, the positive integers n and m and the probabilities γ and ε denote, respectively, the population size, the chromosome length, the crossover rate, and the mutation rate. For simplicity, we assume the population size n to be even. We further assume the crossover rate γ and the mutation rate ε to remain constant over time. We also assume ε to be positive. The GAs that we analyze are generalizations of the canonical GA discussed in, for example, [27, 39]. Like the canonical GA, we assume the use of a binary encoding, that is, chromosomes correspond to bit strings in our GAs. Unlike the canonical GA, we do not assume the use of specific selection and crossover operators. Instead, the GAs that we analyze may use almost any selection operator, such as roulette wheel selection (sometimes referred to as fitness-proportionate selection), tournament selection, or rank selection, and any crossover operator, such as single-point crossover, two-point crossover, or uniform crossover. Furthermore, in the GAs that we analyze, the fitness of a chromosome may depend, either deterministically or stochastically, on the entire population rather than only on the chromosome itself. When using GAs for social modeling, the fitness of a chromosome typically depends on the entire population. This is referred to as state-dependent fitness in [19]. In most studies, GAs that are used for social modeling have the same general form as the GAs that we analyze in this paper.

We now introduce the terminology and the mathematical notation that we use in our analysis. We note that an overview of the mathematical notation is provided in Table 1. There are $\mu = 2^m$ different chromosomes, denoted by $0, \ldots, \mu - 1$. Each chromosome has a unique binary encoding, which is given by a bit string of length m.[1] $\mathcal{C} = \{0, \ldots, \mu - 1\}$ denotes the set of all chromosomes. i and j denote typical chromosomes and take values in $\mathcal{C}$. The following definition introduces the notion of uniform and non-uniform populations.

[1] In this paper, we use a standard binary encoding. Hence, if $m = 2$, chromosomes 0, 1, 2, and 3 have binary encodings 00, 01, 10, and 11, respectively. We emphasize that the use of a standard binary encoding is by no means essential for our analysis. Other binary encoding schemes, such as Gray encoding, can be used as well. This does not require any significant changes in our analysis.

Input: n, m, γ, and ε

1 Initialize the population by randomly setting nm bits to zero or one

2 **repeat**

3 Selection: Apply the selection operator to select n chromosomes from the population (a chromosome may be selected more than once), and use the selected chromosomes as the new population

4 Crossover: Randomly partition the population into $n/2$ pairs of two chromosomes, and apply the crossover operator to each pair of chromosomes with probability γ

5 Mutation: Mutate the population by inverting each bit with probability ε

6 **until** some stopping criterion is satisfied

Figure 1: General form of the genetic algorithms analyzed in this paper.

Definition 1. A population is said to be *uniform* if and only if all n chromosomes in the population are identical. A population is said to be *non-uniform* if and only if some chromosomes in the population are different.

$\mathcal{U}$ denotes the set of all uniform populations. Obviously, since there are μ different chromosomes, there are also μ different uniform populations, that is, $|\mathcal{U}| = \mu$. $u(i) \in \mathcal{U}$ denotes the uniform population consisting of n times chromosome i. $\delta(i, j)$ denotes the Hamming distance between chromosomes i and j, that is, the number of corresponding bits in the binary encodings of i and j that are different. $\mathcal{G}(i)$ denotes the set of all chromosomes that have the same binary encoding as chromosome i except that one bit has been changed from one into zero. Conversely, $\mathcal{H}(i)$ denotes the set of all chromosomes that have the same binary encoding as chromosome i except that one bit has been changed from zero into one. In mathematical notation,

$$\mathcal{G}(i) = \{j \mid j < i \text{ and } \delta(i, j) = 1\}$$
$$\mathcal{H}(i) = \{j \mid j > i \text{ and } \delta(i, j) = 1\}.$$

Notice that $j \in \mathcal{G}(i)$ if and only if $i \in \mathcal{H}(j)$. There are

$$\nu = \mu m/2 = m2^{m-1}$$

combinations of two chromosomes i and j such that $\delta(i, j) = 1$, that is, such that the binary encodings of i and j differ by exactly one bit. k and k' denote indices that take values in $\{1, \ldots, \nu\}$. $\widetilde{\mathcal{V}}$ denotes the set of all populations in which there are exactly two different chromosomes and in which the binary encodings of these chromosomes differ by exactly one bit.

There are

$$\xi = |\widetilde{\mathcal{V}}| = \nu(n-1) = (n-1)m2^{m-1}$$

such populations. (The order of the chromosomes within a population has no effect on the behavior of a GA. Populations consisting of the same chromosomes in different orders are therefore considered identical.) $\mathcal{V}$ denotes the set that is obtained by adding the uniform populations to $\widetilde{\mathcal{V}}$, that is, $\mathcal{V} = \widetilde{\mathcal{V}} \cup \mathcal{U}$. For i and j such that $\delta(i,j) = 1$ and for $\lambda \in \{0,\ldots,n\}$, $v(i,j,\lambda) \in \mathcal{V}$ denotes the population consisting of λ times chromosome i and $n-\lambda$ times chromosome j. Notice that $v(i,j,\lambda) = v(j,i,n-\lambda)$ and that $v(i,j,0) = u(j)$ and $v(i,j,n) = u(i)$. $\mathcal{W}$ denotes the set of all possible populations. As shown in [42, Lemma 1] and [19], the number of possible populations equals

$$|\mathcal{W}| = \binom{n+\mu-1}{\mu-1} = \frac{(n+\mu-1)!}{n!(\mu-1)!}.$$

(Again, populations consisting of the same chromosomes in different orders are considered identical.) For $t \in \{0,1,\ldots\}$, the random variable $W_t \in \mathcal{W}$ denotes the population at the beginning of iteration t of a GA. For i and j such that $\delta(i,j) = 1$ and for $\lambda \in \{1,\ldots,n-1\}$ and $\lambda' \in \{0,\ldots,n\}$, $\pi(i,j,\lambda,\lambda')$ denotes the limit as the mutation rate ε approaches zero of the probability that population $v(i,j,\lambda)$ is turned into population $v(i,j,\lambda')$ in a single iteration of a GA. In mathematical notation,

$$\pi(i,j,\lambda,\lambda') = \lim_{\varepsilon\to 0} \Pr(W_{t+1} = v(i,j,\lambda') \,|\, W_t = v(i,j,\lambda)) \tag{1}$$

where $t \in \{0,1,\ldots\}$. Because the binary encodings of the chromosomes i and j differ by only one bit, the crossover operator has no effect on $\pi(i,j,\lambda,\lambda')$. Moreover, because ε approaches zero, the mutation operator has no effect on $\pi(i,j,\lambda,\lambda')$ either. $\pi(i,j,\lambda,\lambda')$ therefore equals the probability that the selection operator turns population $v(i,j,\lambda)$ into population $v(i,j,\lambda')$ in a single iteration of a GA.

The following definition introduces the notion of almost uniform populations.

Definition 2. A non-uniform population $w \in \mathcal{W} \setminus \mathcal{U}$ is said to be *almost uniform* if and only if

$$\lim_{\varepsilon\to 0} \Pr(W_{t+N} = u \,|\, W_t = w) > 0$$

for all $t \in \{0,1,\ldots\}$, some finite positive integer N, and some $u \in \mathcal{U}$.

Hence, a non-uniform population is almost uniform if and only if no mutation is required to go from the non-uniform population to some uniform population. We note that in many cases

Table 1: Overview of the mathematical notation.

$\mathcal{C}$	Set of all chromosomes
$\mathcal{G}(i)$	Set of all chromosomes that have the same binary encoding as chromosome i except that one bit has been changed from one into zero
$\mathcal{H}(i)$	Set of all chromosomes that have the same binary encoding as chromosome i except that one bit has been changed from zero into one
m	Chromosome length
n	Population size
$\bar{q}(w)$	Long-run probability of population w
$\hat{q}(w)$	Long-run limit probability of population w
$\hat{\mathbf{q}}$	Long-run limit distribution
$\mathcal{U}$	Set of all uniform populations
$u(i)$	Uniform population consisting of n times chromosome i
$\mathcal{V}$	Set of all populations in which there are at most two different chromosomes and in which the binary encodings of chromosomes differ by at most one bit
$v(i, j, \lambda)$	Population consisting of λ times chromosome i and $n - \lambda$ times chromosome j
$\mathcal{W}$	Set of all populations
W_t	Population at the beginning of iteration t of a GA
γ	Crossover rate
$\delta(i, j)$	Hamming distance between chromosomes i and j
ε	Mutation rate
μ	Number of different chromosomes Number of uniform populations
ν	Number of combinations of two chromosomes whose binary encodings differ by exactly one bit
ξ	Number of populations in which there are exactly two different chromosomes and in which the binary encodings of chromosomes differ by at most one bit
$\pi(i, j, \lambda, \lambda')$	Probability that the selection operator turns population $v(i, j, \lambda)$ into population $v(i, j, \lambda')$ in a single iteration of a GA

all non-uniform populations are almost uniform. For example, if a GA uses roulette wheel selection or tournament selection, the selection operator can turn any non-uniform population into a uniform population in a single iteration and, consequently, all non-uniform populations are almost uniform.

The following two definitions introduce the notion of a connection from one chromosome to another.

Definition 3. A *direct connection* from chromosome i to chromosome j is said to exist if and only if $\delta(i, j) = 1$ and

$$\lim_{\varepsilon \to 0} \Pr(W_{t+N} = u(j) \,|\, W_t = v(i, j, n-1)) > 0$$

for all $t \in \{0, 1, \ldots\}$ and some finite positive integer N.

Definition 4. A *connection* from chromosome i to chromosome j is said to exist if and only if there exists a sequence $(i_1, \ldots, i_N)$ such that $i_1, \ldots, i_N \in \mathcal{C}$, $i_1 = i$, $i_N = j$, and i_M is directly connected to i_{M+1} for all $M \in \{1, \ldots, N-1\}$.

Definition 3 states that there is a direct connection from chromosome i to chromosome j if and only if the minimum number of mutations required to go from uniform population $u(i)$ to uniform population $u(j)$ is one. We note that in many cases all chromosomes i and j such that $\delta(i, j) = 1$ have mutual direct connections. This is for example the case if a GA uses roulette wheel selection and the fitness of a chromosome is always positive. Definition 4 states that there is a connection from chromosome i to chromosome j if and only if there is a sequence of chromosomes starting at i and ending at j such that each chromosome in the sequence is directly connected to its successor. Clearly, if all chromosomes i and j such that $\delta(i, j) = 1$ have mutual direct connections, then each chromosome is connected to all other chromosomes.

It is well-known that the population in the current iteration of a GA has no effect on the behavior of the GA in the long run (e.g., [19, 42]). More specifically, the population an infinite number of iterations in the future is statistically independent of the population in the current iteration. The following lemma states this result in a formal way.

Lemma 1. *For each population $w \in \mathcal{W}$, there exists a* long-run probability $\bar{q}(w)$ *such that*

$$\lim_{N \to \infty} \Pr(W_{t+N} = w \,|\, W_t = w_t) = \bar{q}(w) \tag{2}$$

for all $t \in \{0, 1, \ldots\}$ and all $w_t \in \mathcal{W}$.

Proof. See the appendix.

In our analysis, we are concerned with the long-run behavior of GAs in the limit as the mutation rate ε approaches zero. We therefore use the following definition.

Definition 5. For $w \in \mathcal{W}$, $\hat{q}(w) = \lim_{\varepsilon \to 0} \bar{q}(w)$ is called the *long-run limit probability* of population w.

We now introduce the vectors and matrices that we need to state the main result of our analysis. We first note that throughout this paper vectors and matrices are represented by, respectively, bold lowercase and bold uppercase letters and that the transpose of a matrix $\mathbf{X}$ is written as $\mathbf{X}^{\mathrm{T}}$. $\mathbf{I}_N$ denotes an identity matrix of order $N \times N$, and $\mathbf{0}_{M \times N}$ and $\mathbf{1}_{M \times N}$ denote matrices of order $M \times N$ in which all elements are equal to, respectively, zero and one. We simply write $\mathbf{I}$, $\mathbf{0}$, or $\mathbf{1}$ when the order of a matrix is clear from the context. $\mathbf{g} = [g_k]$ and $\mathbf{h} = [h_k]$ denote vectors of length ν that satisfy

$$\forall k : g_k, h_k \in \mathcal{C}$$

$$\forall k : h_k \in \mathcal{H}(g_k)$$

$$\forall k, k' : k \neq k' \Rightarrow (g_k, h_k) \neq (g_{k'}, h_{k'}).$$

Hence, for each k, (g_k, h_k) denotes a combination of two chromosomes such that the binary encodings of the chromosomes differ by exactly one bit. $\mathbf{g}$ and $\mathbf{h}$ together contain all such combinations of two chromosomes. $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ denote matrices of order $\mu \times \xi$, $\xi \times \mu$, $\xi \times \xi$, and $\mu \times \mu$, respectively. Matrix $\mathbf{A}$ is given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}(0,1) & \cdots & \mathbf{a}(0,\nu) \\ \vdots & \ddots & \vdots \\ \mathbf{a}(\mu-1,1) & \cdots & \mathbf{a}(\mu-1,\nu) \end{bmatrix} \tag{3}$$

where

$$\mathbf{a}(i,k) = \begin{cases} \tilde{\mathbf{a}}_{\mathbf{1}}, & \text{if } g_k = i \\ \tilde{\mathbf{a}}_{\mathbf{2}}, & \text{if } h_k = i \\ \mathbf{0}_{1 \times (n-1)}, & \text{otherwise} \end{cases} \tag{4}$$

and

$$\tilde{\mathbf{a}}_{\mathbf{1}} = \begin{bmatrix} \mathbf{0}_{1 \times (n-2)} & 1 \end{bmatrix} \qquad \tilde{\mathbf{a}}_{\mathbf{2}} = \begin{bmatrix} 1 & \mathbf{0}_{1 \times (n-2)} \end{bmatrix}. \tag{5}$$

Matrix $\mathbf{B}$ is given by

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}(1,0) & \cdots & \mathbf{b}(1,\mu-1) \\ \vdots & \ddots & \vdots \\ \mathbf{b}(\nu,0) & \cdots & \mathbf{b}(\nu,\mu-1) \end{bmatrix} \tag{6}$$

where

$$\mathbf{b}(k,i) = \begin{cases} \tilde{\mathbf{b}}(g_k,h_k,n), & \text{if } g_k = i \\ \tilde{\mathbf{b}}(g_k,h_k,0), & \text{if } h_k = i \\ \mathbf{0}_{(n-1)\times 1}, & \text{otherwise} \end{cases} \tag{7}$$

and

$$\tilde{\mathbf{b}}(i,j,\lambda) = \begin{bmatrix} \pi(i,j,1,\lambda) \\ \vdots \\ \pi(i,j,n-1,\lambda) \end{bmatrix}. \tag{8}$$

Matrix $\mathbf{C}$ is given by

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}(1,1) & \cdots & \mathbf{C}(1,\nu) \\ \vdots & \ddots & \vdots \\ \mathbf{C}(\nu,1) & \cdots & \mathbf{C}(\nu,\nu) \end{bmatrix} \tag{9}$$

where

$$\mathbf{C}(k,k') = \begin{cases} \widetilde{\mathbf{C}}(g_k,h_k), & \text{if } k = k' \\ \mathbf{0}_{(n-1)\times(n-1)}, & \text{otherwise} \end{cases} \tag{10}$$

and

$$\widetilde{\mathbf{C}}(i,j) = \begin{bmatrix} \pi(i,j,1,1) & \cdots & \pi(i,j,1,n-1) \\ \vdots & \ddots & \vdots \\ \pi(i,j,n-1,1) & \cdots & \pi(i,j,n-1,n-1) \end{bmatrix}. \tag{11}$$

Matrix $\mathbf{D}$ is obtained from $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ and is given by

$$\mathbf{D} = \mathbf{A}(\mathbf{I}-\mathbf{C})^{-1}\mathbf{B} - m\mathbf{I}. \tag{12}$$

The following theorem states the main result of our analysis.

Theorem 1. *Let all non-uniform populations be almost uniform, and let each chromosome in $\mathcal{C}$ be connected to all other chromosomes in $\mathcal{C}$. Then, (i) all non-uniform populations have a long-run limit probability of zero, that is, $\hat{q}(w) = 0$ for all $w \in \mathcal{W} \setminus \mathcal{U}$, and (ii) the* long-run

limit distribution $\hat{\mathbf{q}} = \begin{bmatrix} \hat{q}(u(0)) & \cdots & \hat{q}(u(\mu-1)) \end{bmatrix}$ *satisfies*

$$\hat{\mathbf{q}}\mathbf{D} = \mathbf{0} \tag{13}$$

$$\hat{\mathbf{q}}\mathbf{1} = 1 \tag{14}$$

which has a unique solution.

Proof. See the appendix.

There are three comments that we would like to make on the above theorem. First, the result that under certain assumptions non-uniform populations have a long-run limit probability of zero is not new. A similar result can be found in [19, Proposition 4.2.1]. Second, under the assumptions of the theorem, the long-run limit probability of a population does not depend on the crossover rate γ. This is a quite remarkable result that, to the best of our knowledge, has not been reported before in the theoretical literature on GAs. It indicates that in the limit as the mutation rate ε approaches zero γ has no effect on the long-run behavior of a GA. Third, the theorem can be used to calculate the long-run limit distribution $\hat{\mathbf{q}}$ only if the probabilities $\pi(i, j, \lambda, \lambda')$ defined in (1) can be calculated for all i and all j such that $\delta(i, j) = 1$ and for all $\lambda \in \{1, \ldots, n-1\}$ and all $\lambda' \in \{0, \ldots, n\}$. Whether this is possible depends on the way in which the fitness of a chromosome is determined and on the selection operator that is used. This in turn depends heavily on the specific problem that one wants to model using a GA. Because of the dependence on the problem to be modeled, we cannot provide any general results for the calculation of the probabilities $\pi(i, j, \lambda, \lambda')$. In Section 4, however, we demonstrate how the probabilities $\pi(i, j, \lambda, \lambda')$ can be calculated for a GA that is similar to the GA used by Axelrod in his seminal paper on GA modeling [12].

3 Algorithm

In this section, we present an algorithm for calculating the long-run limit distribution $\hat{\mathbf{q}}$. The algorithm is based on Theorem 1. Like Theorem 1, it assumes that all non-uniform populations are almost uniform and that each chromosome in $\mathcal{C}$ is connected to all other chromosomes in $\mathcal{C}$. It also assumes that the probabilities $\pi(i, j, \lambda, \lambda')$ defined in (1) can be calculated for all i and all j such that $\delta(i, j) = 1$ and for all $\lambda \in \{1, \ldots, n-1\}$ and all $\lambda' \in \{0, \ldots, n\}$.

The most straightforward approach to calculating the long-run limit distribution $\hat{\mathbf{q}}$ would be to start with calculating the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ using (3)–(11). Matrix $\mathbf{D}$ would then be calculated using (12), which would require solving a linear system. Finally, $\hat{\mathbf{q}}$ would be obtained by solving the linear system given by (13) and (14). Unfortunately, this approach to calculating $\hat{\mathbf{q}}$ requires a lot of computer memory and is therefore infeasible even for problems of only moderate size. Most memory is required for storing matrix $\mathbf{C}$. This matrix has (at most) $\nu(n-1)^2 = (n-1)^2 m 2^{m-1}$ non-zero elements. Clearly, as the population size n and the chromosome length m increase, storing the non-zero elements of $\mathbf{C}$ in a computer's main memory soon becomes infeasible. The algorithm that we propose for calculating $\hat{\mathbf{q}}$ exploits the sparsity of the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ in order to calculate matrix $\mathbf{D}$ in a memory-efficient way. The algorithm does not require the entire matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ to be stored in memory. The algorithm also solves the linear system given by (13) and (14) in a memory-efficient way. This is achieved by exploiting the sparsity of $\mathbf{D}$. The algorithm is shown in Figure 2. We now discuss it in more detail.

We first consider the efficient calculation of matrix $\mathbf{D}$. Let $\widehat{\mathbf{C}} = (\mathbf{I} - \mathbf{C})^{-1}$. Because $\mathbf{C}$ is a block diagonal matrix, $\widehat{\mathbf{C}}$ can be written as

$$\widehat{\mathbf{C}} = \begin{bmatrix} \widehat{\mathbf{C}}(1,1) & \cdots & \widehat{\mathbf{C}}(1,\nu) \\ \vdots & \ddots & \vdots \\ \widehat{\mathbf{C}}(\nu,1) & \cdots & \widehat{\mathbf{C}}(\nu,\nu) \end{bmatrix}$$

where

$$\widehat{\mathbf{C}}(k,k') = \begin{cases} (\mathbf{I} - \widetilde{\mathbf{C}}(g_k, h_k))^{-1}, & \text{if } k = k' \\ \mathbf{0}_{(n-1)\times(n-1)}, & \text{otherwise.} \end{cases}$$

Hence, $\widehat{\mathbf{C}}$ is a block diagonal matrix too. Let $\mathbf{D}$ be written as

$$\mathbf{D} = \begin{bmatrix} d(0,0) & \cdots & d(0,\mu-1) \\ \vdots & \ddots & \vdots \\ d(\mu-1,0) & \cdots & d(\mu-1,\mu-1) \end{bmatrix}.$$

Input: n, m, $\hat{\mathbf{q}}_\mathbf{0}$, and ω

Output: $\hat{\mathbf{q}}$

// Calculation of $\mathbf{D} = \left[d(i,j)\right]$
// Only non-zero elements of $\mathbf{D}$ should be stored
$\mu \leftarrow 2^m$
$\mathbf{D} \leftarrow -m\mathbf{I}_\mu$
$\tilde{\mathbf{a}}_\mathbf{1} \leftarrow \tilde{\mathbf{a}}_\mathbf{1}$ given by (5)
$\tilde{\mathbf{a}}_\mathbf{2} \leftarrow \tilde{\mathbf{a}}_\mathbf{2}$ given by (5)
for $i \leftarrow 0$ **to** $\mu - 1$ **do**
    for all $j \in \mathcal{H}(i)$ **do**
        $\tilde{\mathbf{b}} \leftarrow \tilde{\mathbf{b}}(i,j,n)$ given by (8)
        $\widetilde{\mathbf{C}} \leftarrow \widetilde{\mathbf{C}}(i,j)$ given by (11)
        $\tilde{\mathbf{e}} \leftarrow (\mathbf{I} - \widetilde{\mathbf{C}})^{-1}\tilde{\mathbf{b}}$ // Use, e.g., Gaussian elimination
        $d(i,i) \leftarrow d(i,i) + \tilde{\mathbf{a}}_\mathbf{1}\tilde{\mathbf{e}}$
        $d(j,j) \leftarrow d(j,j) + 1 - \tilde{\mathbf{a}}_\mathbf{2}\tilde{\mathbf{e}}$
        $d(i,j) \leftarrow 1 - \tilde{\mathbf{a}}_\mathbf{1}\tilde{\mathbf{e}}$
        $d(j,i) \leftarrow \tilde{\mathbf{a}}_\mathbf{2}\tilde{\mathbf{e}}$
    end for
end for
// Calculation of $\hat{\mathbf{q}} = \left[\hat{q}(u(i))\right]$
// The linear system given by (13) and (14) will be solved using successive overrelaxation
$\hat{\mathbf{q}} \leftarrow \hat{\mathbf{q}}_\mathbf{0}$
repeat
    for $i \leftarrow 0$ **to** $\mu - 1$ **do**
        $\sigma \leftarrow 0$
        for all $j \in \mathcal{G}(i) \cup \mathcal{H}(i)$ **do**
            $\sigma \leftarrow \sigma + \hat{q}(u(j))d(j,i)$
        end for
        $\sigma \leftarrow -\sigma/d(i,i)$
        $\hat{q}(u(i)) \leftarrow (1-\omega)\hat{q}(u(i)) + \omega\sigma$
    end for
until some convergence criterion is satisfied
$\hat{\mathbf{q}} \leftarrow \hat{\mathbf{q}}/(\hat{\mathbf{q}}\mathbf{1})$

Figure 2: Algorithm for calculating the long-run limit distribution of a genetic algorithm.

Taking into account the sparsity of $\mathbf{A}$, $\mathbf{B}$, and $\widehat{\mathbf{C}}$, it can be seen that

$$d(i,j) = \begin{cases} \sum_{i' \in \mathcal{G}(i)} \tilde{\mathbf{a}}_\mathbf{2}\tilde{\mathbf{e}}(i',i,0) + \sum_{i' \in \mathcal{H}(i)} \tilde{\mathbf{a}}_\mathbf{1}\tilde{\mathbf{e}}(i,i',n) - m, & \text{if } i = j \\ \tilde{\mathbf{a}}_\mathbf{2}\tilde{\mathbf{e}}(j,i,n), & \text{if } j \in \mathcal{G}(i) \\ \tilde{\mathbf{a}}_\mathbf{1}\tilde{\mathbf{e}}(i,j,0), & \text{if } j \in \mathcal{H}(i) \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

where

$$\tilde{\mathbf{e}}(i,j,\lambda) = (\mathbf{I} - \widetilde{\mathbf{C}}(i,j))^{-1}\tilde{\mathbf{b}}(i,j,\lambda).$$

This result shows that each non-zero element of $\mathbf{D}$ can be calculated by solving one or more relatively small linear systems, that is, systems of $n-1$ equations and unknowns. Moreover, by calculating the elements of $\mathbf{D}$ one by one, there is no need to store the entire matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ in memory. Solving a linear system of $n-1$ equations and unknowns can be done using standard Gaussian elimination methods. Except for very large values for the population size n, today's computers have sufficient main memory to apply Gaussian elimination methods to such systems. We further note that the amount of computation required for obtaining $\mathbf{D}$ can be reduced by taking into account that

$$\begin{aligned} \tilde{\mathbf{e}}(i,j,0) &= (\mathbf{I} - \widetilde{\mathbf{C}}(i,j))^{-1}\tilde{\mathbf{b}}(i,j,0) \\ &= (\mathbf{I} - \widetilde{\mathbf{C}}(i,j))^{-1}(\mathbf{1} - \textstyle\sum_{\lambda=1}^{n} \tilde{\mathbf{b}}(i,j,\lambda)) \\ &= (\mathbf{I} - \widetilde{\mathbf{C}}(i,j))^{-1}(\mathbf{1} - \widetilde{\mathbf{C}}(i,j)\mathbf{1} - \tilde{\mathbf{b}}(i,j,n)) \\ &= (\mathbf{I} - \widetilde{\mathbf{C}}(i,j))^{-1}(\mathbf{I} - \widetilde{\mathbf{C}}(i,j))\mathbf{1} - \tilde{\mathbf{e}}(i,j,n) \\ &= \mathbf{1} - \tilde{\mathbf{e}}(i,j,n). \end{aligned}$$

Because of this, $d(i,j)$ can be written as

$$d(i,j) = \begin{cases} \sum_{i' \in \mathcal{G}(i)} (1 - \tilde{\mathbf{a}}_\mathbf{2}\tilde{\mathbf{e}}(i',i,n)) + \sum_{i' \in \mathcal{H}(i)} \tilde{\mathbf{a}}_\mathbf{1}\tilde{\mathbf{e}}(i,i',n) - m, & \text{if } i = j \\ \tilde{\mathbf{a}}_\mathbf{2}\tilde{\mathbf{e}}(j,i,n), & \text{if } j \in \mathcal{G}(i) \\ 1 - \tilde{\mathbf{a}}_\mathbf{1}\tilde{\mathbf{e}}(i,j,n), & \text{if } j \in \mathcal{H}(i) \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Using (16) rather than (15) to calculate $\mathbf{D}$ halves the number of linear systems that need to be solved. In the algorithm in Figure 2, the calculation of $\mathbf{D}$ based on (16) is performed between lines 1 and 17.

Matrix $\mathbf{D}$ has $\mu^2 = 2^{2m}$ elements. Consequently, storing all elements of $\mathbf{D}$ in a computer's main memory is possible only if the chromosome length m is not too large. It follows from (15) and (16) that the number of non-zero elements in $\mathbf{D}$ equals $\mu(m+1) = (m+1)2^m$. Hence, $\mathbf{D}$ is a rather sparse matrix and a lot of memory can be saved by storing only its non-zero elements.[2] In addition to the memory efficiency of the way in which $\mathbf{D}$ is stored, one should also pay attention to the memory efficiency of the method that is used to solve the linear system given by (13) and (14). Gaussian elimination and other direct (i.e., non-iterative) methods for solving linear systems generally require that at least a large number of elements of the coefficient matrix, including zero elements, are stored in memory. Consequently, when using such a method to solve the linear system given by (13) and (14), it would not be possible to fully exploit the sparsity of $\mathbf{D}$. Linear systems can also be solved using iterative methods that require only the non-zero elements of the coefficient matrix to be stored in memory. One such method is the method of successive overrelaxation (e.g., [14, 45, 48, 49]). In the algorithm in Figure 2, this method is used to solve the linear system given by (13) and (14) (see lines 18–31 of the algorithm). In addition to an initial guess $\hat{\mathbf{q}}_\mathbf{0}$ for the solution of the linear system, the method of successive overrelaxation also requires a value for the relaxation factor ω. The value of ω, which should be between 0 and 2, may have a large effect on the rate of convergence of the method, and for some values of ω the method may not converge at all. An appropriate value for ω has to be determined experimentally. For $\omega = 1$, the method of successive overrelaxation reduces to the Gauss-Seidel method, which is another iterative method for solving linear systems. We refer to [45] for an in-depth discussion of both the method of successive overrelaxation and a number of alternative methods for solving linear systems similar to the one given by (13) and (14). We further note that the amount of main memory in most of today's computers allows the algorithm in Figure 2 to be run for chromosomes with length m up to about 24 bits.

[2]The non-zero elements of $\mathbf{D}$ can be stored efficiently by using two arrays: a one-dimensional array of size μ for the diagonal elements of $\mathbf{D}$ and a two-dimensional array of size $m \times \mu$ for the non-zero off-diagonal elements of $\mathbf{D}$. The element in the κth row and the ith column of the latter array is used to store $d(j, i)$, where j has the same binary encoding as i except that the κth bit is inverted.

4 Application

In this section, we demonstrate an application of the algorithm presented in the previous section. We study the use of a GA for modeling the evolution of strategies in iterated prisoner's dilemmas (IPDs). The use of GAs in this context was first studied by Axelrod [12] (reprinted in [13]; see also [19, 39]) and after him by many others (e.g., [4, 8–10, 18, 30, 38, 40, 41, 47, 50, 57]). The algorithm presented in the previous section is used to analyze the long-run behavior of our GA. The results of the analysis are compared with results obtained using computer simulations. We emphasize that our primary aim is merely to illustrate the usefulness of the mathematical analysis provided in Section 2 and of the algorithm derived from the analysis in Section 3. It is not our primary aim to provide new insights into the behavior of GAs in the context of IPDs.

4.1 Genetic algorithm modeling in iterated prisoner's dilemmas

The way in which we model the evolution of strategies in IPDs is similar to the way in which this was done by Axelrod [12]. However, Axelrod studied two approaches for modeling the evolution of strategies. In one approach, the fitness of a chromosome is determined by the performance of the chromosome in IPD games against a fixed set of opponents. In the other approach, the fitness of a chromosome is determined by the performance of the chromosome in IPD games against other chromosomes in the population. We restrict our attention to the second approach. This is the approach on which almost all studies after Axelrod's work have focused (an exception is [40]).

We model the evolution of strategies in IPDs using a GA with a population size of $n = 20$ chromosomes. Each chromosome represents a strategy for playing IPD games. Players in IPD games are assumed to choose the action they play, that is, whether they cooperate or defect, based on their own actions and their opponent's actions in the previous τ periods of the game, where τ is referred to as players' memory length. Players are further assumed to play only pure strategies. We use the same binary encoding of strategies as was used by Axelrod [12]. For a description of this encoding, we refer to [12, 13, 19, 39]. Using Axelrod's encoding, the chromosome length m depends on the memory length τ. We consider three memory lengths, 1, 2, and 3 periods, which result in chromosome lengths of, respectively, 6, 20, and 70 bits. In each iteration of the GA, each chromosome in the population plays an IPD game of 151 periods

Table 2: Payoff matrix for a single period of an iterated prisoner's dilemma game. The payoff obtained by the row (column) player is reported first (second).

	Cooperate	Defect
Cooperate	R, R	S, T
Defect	T, S	P, P

against all other chromosomes. In addition, each chromosome also plays a game against itself. The payoff matrix for a single period of an IPD game is shown in Table 2. The payoffs in this matrix must satisfy

$$S < P < R < T$$

and

$$S + T < 2R.$$

The payoff obtained by a chromosome in an IPD game equals the mean payoff obtained by the chromosome in all periods of the game. The fitness f of a chromosome equals the mean payoff obtained by the chromosome in the IPD games that it has played in the current iteration of the GA. Like in Axelrod's work [12], we use sigma scaling (e.g., [39]) to normalize the fitness of a chromosome. The normalized fitness $\tilde{f}$ of a chromosome is given by

$$\tilde{f} = \begin{cases} \max\left(\dfrac{f - \mu_f}{\sigma_f} + 1, 0\right), & \text{if } \sigma_f > 0 \\ 1, & \text{otherwise} \end{cases} \tag{17}$$

where μ_f and σ_f denote, respectively, the mean and the standard deviation of the fitness of the chromosomes in the population. The selection operator that we use is roulette wheel selection. Selection is performed based on the normalized fitness of the chromosomes in the population. The crossover operator that we use is single-point crossover.

4.2 Calculation of the long-run limit distribution of the genetic algorithm

In this subsection, we are concerned with the calculation of the long-run limit distribution of the GA discussed in the previous subsection. To calculate the long-run limit distribution of the GA, we use the algorithm presented in Section 3. This algorithm assumes that the probabilities $\pi(i, j, \lambda, \lambda')$ defined in (1) can be calculated for all i and all j such that $\delta(i, j) = 1$ and for all

$\lambda \in \{1, \ldots, n-1\}$ and all $\lambda' \in \{0, \ldots, n\}$. We now discuss the calculation of the probabilities $\pi(i, j, \lambda, \lambda')$ for our GA. For $i', j' \in \mathcal{C}$, let $\varphi(i', j')$ denote the payoff obtained by chromosome i' in an IPD game against chromosome j'. Suppose that the population in the current iteration of our GA equals $v(i, j, \lambda)$, where i and j satisfy $\delta(i, j) = 1$ and where $\lambda \in \{1, \ldots, n-1\}$. That is, the population in the current iteration of our GA consists of λ times chromosome i and $n - \lambda$ times chromosome j. The fitness f_i of chromosome i is then given by

$$f_i = \frac{\lambda\varphi(i,i) + (n-\lambda)\varphi(i,j)}{n}.$$

Similarly, the fitness f_j of chromosome j is given by

$$f_j = \frac{\lambda\varphi(j,i) + (n-\lambda)\varphi(j,j)}{n}.$$

Furthermore, the mean μ_f and the standard deviation σ_f of the fitness of the chromosomes in the population are equal to, respectively,

$$\mu_f = \frac{\lambda f_i + (n-\lambda) f_j}{n}$$

and

$$\sigma_f = \sqrt{\frac{\lambda(f_i - \mu_f)^2 + (n-\lambda)(f_j - \mu_f)^2}{n}}.$$

The normalized fitness $\tilde{f}_i$ of chromosome i is obtained by substituting f_i, μ_f, and σ_f into (17). The normalized fitness $\tilde{f}_j$ of chromosome j is obtained in a similar way. Let $\tilde{\pi}_i$ and $\tilde{\pi}_j$ denote the probabilities that the roulette wheel selection operator selects, respectively, chromosome i and chromosome j. Obviously, $\tilde{\pi}_i$ and $\tilde{\pi}_j$ equal

$$\tilde{\pi}_i = \frac{\lambda\tilde{f}_i}{\lambda\tilde{f}_i + (n-\lambda)\tilde{f}_j} \qquad\qquad \tilde{\pi}_j = \frac{(n-\lambda)\tilde{f}_j}{\lambda\tilde{f}_i + (n-\lambda)\tilde{f}_j}.$$

$\pi(i, j, \lambda, \lambda')$, where $\lambda' \in \{0, \ldots, n\}$, equals the probability that the roulette wheel selection operator turns population $v(i, j, \lambda)$ into population $v(i, j, \lambda')$ in a single iteration of our GA. Taking into account that the roulette wheel selection operator selects chromosomes independently of each other, it can be seen that $\pi(i, j, \lambda, \lambda')$ equals the probability mass function of a binomial distribution and is given by

$$\pi(i, j, \lambda, \lambda') = \binom{n}{\lambda'} {\tilde{\pi}_i}^{\lambda'} {\tilde{\pi}_j}^{n-\lambda'}$$

where the binomial coefficient $\binom{n}{\lambda'}$ is defined as

$$\binom{n}{\lambda'} = \frac{n!}{\lambda'!(n-\lambda')!}.$$

The algorithm presented in Section 3 also assumes that all non-uniform populations are almost uniform and that each chromosome in $\mathcal{C}$ is connected to all other chromosomes in $\mathcal{C}$. Because of the use of roulette wheel selection, the assumption that all non-uniform populations are almost uniform is satisfied. The assumption that each chromosome in $\mathcal{C}$ is connected to all other chromosomes in $\mathcal{C}$ is satisfied if and only if matrix $\mathbf{D}$ calculated in lines 1–17 of the algorithm in Figure 2 is irreducible. ($\mathbf{D} = \Big[d(i,j)\Big]$ is said to be irreducible if and only if there does not exist a non-empty set of chromosomes $\widetilde{\mathcal{C}} \subset \mathcal{C}$ such that $d(i,j) = 0$ for all $i \in \widetilde{\mathcal{C}}$ and all $j \in \mathcal{C} \setminus \widetilde{\mathcal{C}}$.) For the particular values that we use for the parameters S, P, R, T, and τ (see the next subsection), $\mathbf{D}$ turns out to be irreducible. Hence, the assumption that each chromosome in $\mathcal{C}$ is connected to all other chromosomes in $\mathcal{C}$ is satisfied.

4.3 Analysis of the long-run behavior of the genetic algorithm

In this subsection, we analyze the long-run behavior of our GA for the prisoner's dilemma payoffs $S = 0$, $P = 1$, $R = 3$, and $T = 5$. These are the same payoffs as were used by Axelrod [12] (see also [11]) and by many others. The analysis is performed using the algorithm presented in Section 3. The use of this algorithm to analyze the long-run behavior of our GA was discussed in the previous subsection. We compare the results obtained using the algorithm with results obtained using computer simulations.[3]

The long-run limit distribution for a memory length of $\tau = 1$ period is shown in Figure 3 (in dark grey). The distribution was calculated using the algorithm from Section 3. As mentioned before, $\tau = 1$ results in a chromosome length of $m = 6$ bits. This implies that there are $\mu = 2^m = 64$ different chromosomes and, as a consequence, that there are 64 different uniform populations. The long-run limit distribution is a probability distribution over these populations. As can be seen in Figure 3, the long-run limit distribution spreads most of its mass over approximately fifteen populations. It puts almost no mass on the remaining populations. Since all

[3]The software used to obtain the results reported in this subsection is available online at `http://www.ludowaltman.nl/ga_analysis/`. The software runs in MATLAB and has been written partly in the MATLAB programming language and partly in the C programming language.

chromosomes in a uniform population are identical and represent the same strategy, the long-run limit distribution can be used to determine the long-run limit probability that a particular strategy is played. However, when doing so, it should be noted that there is some redundancy in the binary encoding of strategies that we use (as was already pointed out by Axelrod [12]). Due to this redundancy, it is possible that different chromosomes represent the same strategy. Some strategies can be encoded in two or three different ways, and the strategies *always cooperate* and *always defect* can even be encoded in twelve different ways. Taking into account the redundancy in the encoding, we have calculated the long-run limit probabilities of all possible strategies. The six strategies with the highest long-run limit probability are reported in Table 3. Together, these strategies have a long-run limit probability of almost 0.95. The remaining strategies all have very low long-run limit probabilities. It is sometimes claimed (e.g., [11, 12]) that a very effective strategy for playing IPD games is the *tit for tat* strategy, which is the strategy of cooperating in the first period and repeating the opponent's previous action thereafter. The results reported in Table 3 do not really support this claim. As can be seen in the table, the *always defect* strategy has by far the highest long-run limit probability. In the long run, this strategy is played about 43% of the time. The *tit for tat* strategy has a long-run limit probability of no more than 0.14. This is even slightly less than the long-run limit probability of another cooperative strategy, namely the strategy that keeps cooperating until the opponent defects and then keeps defecting forever.

In order to check the correctness of the algorithm presented in Section 3, we have also used computer simulations to analyze the long-run behavior of our GA. Like above, we first focus on the behavior of the GA for a memory length of $\tau = 1$ period. We performed 500 runs of the GA. The crossover rate was set to $\gamma = 1.0$, and the mutation rate was set to $\varepsilon = 10^{-5}$. Because of the very small value of ε, the simulation results should be similar to the results obtained using the algorithm from Section 3. (Recall that the latter results hold in the limit as ε approaches zero.) Each run of the GA lasted $2 \cdot 10^5$ iterations. This seemed sufficient for the GA to reach its steady state. After the last iteration of a GA run, we almost always observed that the population was uniform. Based on the 500 GA runs that we had performed, we estimated for each uniform population the probability of observing that population at the end of a GA run. In this way, we obtained a probability distribution over the uniform populations. This distribution is shown in Figure 3 (in light grey). Figure 3 allows us to compare the distribution with the

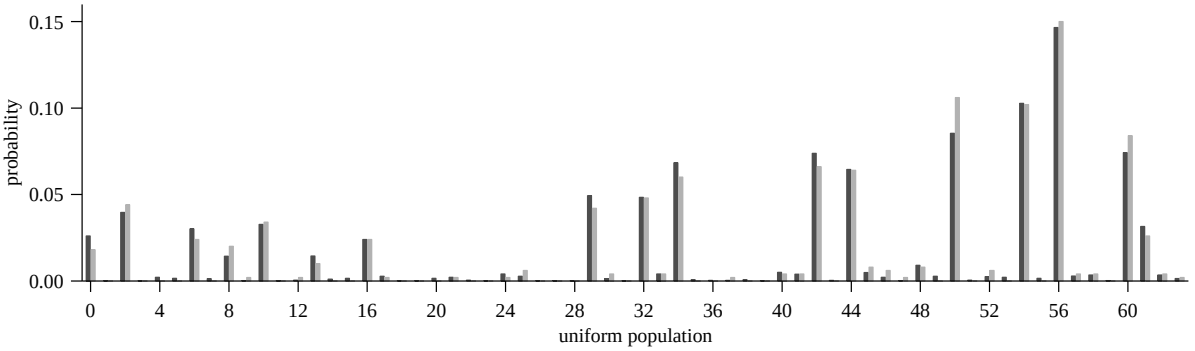

Figure 3: The long-run limit distribution calculated using the algorithm presented in Section 3 (in dark grey) and a probability distribution over the uniform populations estimated using computer simulations (in light grey). The memory length τ equals 1. On the horizontal axis, integers between 0 and 63 are used to represent the uniform populations. Integer i represents the uniform population consisting of 20 times chromosome i.

Table 3: The six strategies with the highest long-run limit probability (reported in the first column). The memory length τ equals 1.

Prob.	Strategy	Chromosomes
0.430	Always defect	0, 2, 8, 10, 16, 24, 32, 34, 40, 42, 48, 50
0.147	Start cooperating; cooperate if and only if both you and your opponent cooperated in the previous period	56
0.139	Start cooperating; cooperate if and only if your opponent cooperated in the previous period (tit for tat)	44, 60
0.133	Start defecting; cooperate if and only if you and your opponent played different actions in the previous period	6, 54
0.051	Start cooperating; cooperate unless you cooperated in the previous period and your opponent did not	13, 45, 61
0.049	Start defecting; cooperate unless you cooperated in the previous period and your opponent did not	29

long-run limit distribution calculated using the algorithm from Section 3. It can be seen that the two distributions are very similar. This confirms the correctness of the algorithm presented in Section 3.

In order to examine to what extent our GA results in the evolution of cooperative strategies, we now focus on the long-run mean fitness, that is, the mean fitness of a chromosome after a large number of iterations of the GA. For various values of the memory length τ, the crossover rate γ, and the mutation rate ε, the long-run mean fitness estimated using computer simulations is reported in Table 4. The associated 95% confidence interval is also provided in the table. The simulation results for $\tau = 1$ are based on 500 runs of the GA, and the simulation results for $\tau = 2$ and $\tau = 3$ are based on 200 runs. Each run lasted $2 \cdot 10^5$ iterations. The long-run mean fitness was estimated by taking the average over all GA runs of the mean fitness of a chromosome at the end of a run. In the limit as ε approaches zero, the long-run mean fitness can be calculated exactly and does not depend on γ. The calculation of the long-run mean fitness is based on the long-run limit distribution of the GA, which can be obtained using the algorithm presented in Section 3. For $\tau = 1$ and $\tau = 2$, the long-run mean fitness in the limit as ε approaches zero is reported in Table 4. For $\tau = 3$, we cannot calculate the long-run limit distribution of the GA and we therefore do not know the long-run mean fitness in the limit as ε approaches zero. Calculating the long-run limit distribution of the GA is impossible for $\tau = 3$ because the chromosome length equals $m = 70$ bits and because for such a chromosome length storing the long-run limit distribution requires a prohibitive amount of computer memory.

Based on the results in Table 4, a number of observations can be made. First, for $\tau = 1$ and $\tau = 2$, the results obtained for $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-5}$ turn out to be very similar to the results obtained for $\varepsilon \to 0$. This again confirms the correctness of the algorithm presented in Section 3. Second, for $\tau = 1$, we find that the results are quite sensitive to the value of ε. Studies on GA modeling sometimes report that the long-run behavior of a GA is relatively insensitive to the value of ε. Our results demonstrate that this need not always be the case. Third, for small values of ε, it can be seen that increasing τ leads to a higher long-run mean fitness and, hence, to more cooperation. The evolution of cooperative strategies in IPD games therefore seems more likely when players have longer memory lengths. Finally, it can be observed that the value of γ has no significant effect on our results. This is in line with the mathematical analysis provided in Section 2. The mathematical analysis implies that for $\varepsilon \to 0$ the long-run mean fitness is

independent of γ. The results in Table 4 indicate that this is the case not only for $\varepsilon \to 0$ but more generally.

5 Conclusions

In this paper, we have presented a mathematical analysis of the long-run behavior of GAs that are used for modeling social phenomena. Under the assumption of a positive but infinitely small mutation rate, the analysis provides a full characterization of the long-run behavior of GAs with a binary encoding. Based on the analysis, we have derived an algorithm for calculating the long-run behavior of GAs. In an economic context, the algorithm can for example be used to determine whether convergence to an equilibrium will take place and, if so, what kind of equilibrium will emerge. Compared with computer simulations, the main advantage of the algorithm that we have derived is that it calculates the long-run behavior of GAs exactly. Computer simulations only estimate the long-run behavior of GAs.

To demonstrate the usefulness of our mathematical analysis, we have replicated a well-known study by Axelrod in which a GA is used to model the evolution of strategies in iterated prisoner's dilemmas [12]. We have used both our exact algorithm and computer simulations to replicate Axelrod's study. By comparing the results of the two approaches, we have confirmed the correctness of our algorithm. We have also obtained some interesting new insights. For example, when players have a memory length of one period, the *tit for tat* strategy turns out to be less important than is sometimes claimed (e.g., [11, 12]). In the long run, the strategy is played only 14% of the time. Another finding is that the long-run behavior of a GA can be quite sensitive to the value of the mutation rate. We regard this as a serious problem, since the value of the mutation rate is typically chosen in a fairly arbitrary way without any empirical justification (see also [19]).

The mathematical analysis that we have presented also reveals that if the mutation rate is infinitely small the crossover rate has no effect on the long-run behavior of a GA. This remarkable result is perfectly in line with the simulation results that we have reported in Section 4. For various values of the mutation rate, the simulation results show no significant effect of the crossover rate on the long-run behavior of a GA. Hence, when GAs are used for modeling social phenomena, the crossover rate seems to be a rather unimportant parameter, at least when the

Table 4: Estimated long-run mean fitness and associated 95% confidence interval for various values of the memory length τ, the crossover rate γ, and the mutation rate ε. For $\varepsilon \to 0$, the long-run mean fitness has been calculated exactly.

	$\tau = 1$			$\tau = 2$			$\tau = 3$		
	$\gamma = 0.0$	$\gamma = 0.5$	$\gamma = 1.0$	$\gamma = 0.0$	$\gamma = 0.5$	$\gamma = 1.0$	$\gamma = 0.0$	$\gamma = 0.5$	$\gamma = 1.0$
$\varepsilon = 10^{-2}$	2.76 ± 0.05	2.71 ± 0.05	2.79 ± 0.04	2.64 ± 0.08	2.72 ± 0.07	2.67 ± 0.07	2.67 ± 0.06	2.64 ± 0.07	2.70 ± 0.06
$\varepsilon = 10^{-3}$	2.23 ± 0.08	2.24 ± 0.08	2.25 ± 0.08	2.34 ± 0.12	2.41 ± 0.11	2.38 ± 0.11	2.55 ± 0.09	2.60 ± 0.09	2.59 ± 0.08
$\varepsilon = 10^{-4}$	1.93 ± 0.09	1.94 ± 0.09	1.90 ± 0.09	2.25 ± 0.12	2.24 ± 0.12	2.32 ± 0.12	2.57 ± 0.09	2.53 ± 0.09	2.50 ± 0.09
$\varepsilon = 10^{-5}$	1.85 ± 0.09	1.81 ± 0.09	1.85 ± 0.09	2.28 ± 0.12	2.31 ± 0.11	2.22 ± 0.12	2.58 ± 0.09	2.44 ± 0.10	2.44 ± 0.10
$\varepsilon \to 0$	1.84	1.84	1.84	2.29	2.29	2.29	?	?	?

focus is on the long run (for the short run, see [47]). It seems likely that in many cases leaving out the crossover operator altogether has no significant effect on the long-run behavior of a GA. Interestingly, leaving out the crossover operator brings GAs quite close to well-known models in evolutionary game theory, such as those studied in [31, 52].

Finally, we note that an analysis such as the one presented in this paper can be performed not only for GAs with a binary encoding but also for other types of evolutionary algorithms. From a modeling point of view, a binary encoding in many cases has the disadvantage that it lacks a clear interpretation (e.g., [19]). The use of a binary encoding can therefore be difficult to justify and may even lead to artifacts (as suggested in [55]). Probably for these reasons, some researchers use evolutionary algorithms without a binary encoding (e.g., [28, 33]). The analysis presented in this paper then does not directly apply. However, when the action space of agents is assumed discrete, the long-run behavior of evolutionary algorithms without a binary encoding can still be analyzed in a similar way as we have done in this paper, namely by relying on mathematical results provided by Freidlin and Wentzell [22]. This indicates that our approach is quite general and can be adapted relatively easily to other types of evolutionary algorithms.

Appendix

In this appendix, we prove the mathematical results presented in Section 2. Before proving the results, we first provide some definitions and lemmas on Markov chains.

Definition 6. A collection of random variables $\{X_t\}$, where the index t takes values in $\{0, 1, \ldots\}$ and where $X_0, X_1, \ldots$ take values in a finite set $\mathcal{X}$, is called a *finite discrete-time Markov chain* if

$$\Pr(X_{t+1} = x_{t+1} \,|\, X_t = x_t) = \Pr(X_{t+1} = x_{t+1} \,|\, X_t = x_t, \ldots, X_0 = x_0)$$

for all t and all $x_0, \ldots, x_{t+1} \in \mathcal{X}$. The elements of $\mathcal{X}$ are called the *states* of the Markov chain. $\mathcal{X}$ is called the *state space* of the Markov chain.

Definition 7. A finite discrete-time Markov chain $\{X_t\}$ is said to be *time-homogeneous* if

$$\Pr(X_{t+1} = x_{t+1} \,|\, X_t = x_t) = p(x_t, x_{t+1})$$

for all t, all $x_t, x_{t+1} \in \mathcal{X}$, and some function $p : \mathcal{X}^2 \to [0, 1]$ that does not depend on t. For $x, x' \in \mathcal{X}$, the probability $p(x, x')$ is called the *transition probability* from state x to state x'.

The matrix

$$\mathbf{P} = \Big[p(x, x')\Big]_{x,x' \in \mathcal{X}}$$

is called the *transition matrix* of the Markov chain.

In the remainder of this appendix, the term Markov chain always refers to a finite discrete-time Markov chain that is time-homogeneous.

Definition 8. Consider a Markov chain $\{X_t\}$. A row vector $\bar{\mathbf{p}} = [\bar{p}(x)]_{x \in \mathcal{X}}$ that satisfies

$$\bar{\mathbf{p}}\mathbf{P} = \bar{\mathbf{p}}$$

$$\bar{\mathbf{p}}\mathbf{1} = 1$$

is called a *stationary distribution* of the Markov chain. For $x \in \mathcal{X}$, the probability $\bar{p}(x)$ is called the *stationary probability* of state x.

Definition 9. A Markov chain $\{X_t\}$ is said to be *irreducible* if for each $x, x' \in \mathcal{X}$ there exists a positive integer N such that $\Pr(X_{t+N} = x' \,|\, X_t = x) > 0$.

Lemma 2. *If a Markov chain $\{X_t\}$ is irreducible, it has a unique stationary distribution $\bar{\mathbf{p}}$.*

Proof. See, for example, [48, Th. 2.3.3].

Definition 10. An irreducible Markov chain $\{X_t\}$ is said to be *aperiodic* if for each $x \in \mathcal{X}$ there exists a positive integer N such that $\Pr(X_{t+M} = x \,|\, X_t = x) > 0$ for all integers $M \geq N$.

Lemma 3. *If a Markov chain $\{X_t\}$ is irreducible and aperiodic, then*

$$\lim_{t \to \infty} \Pr(X_t = x \,|\, X_0 = x_0) = \bar{p}(x)$$

for all $x, x_0 \in \mathcal{X}$.

Proof. See, for example, [48, Th. 2.3.1 and Lemma 2.3.2].

Lemma 4. *Let a Markov chain $\{X_t\}$ be irreducible. Let $\mathcal{Y} \subset \mathcal{X}$ and $\mathcal{Y} \neq \varnothing$. Let*

$$\mathbf{T} = \Big[p(x, x')\Big]_{x,x' \in \mathcal{Y}} \qquad \mathbf{U} = \Big[p(x, x')\Big]_{x \in \mathcal{Y}, x' \in \mathcal{X} \setminus \mathcal{Y}}$$

$$\mathbf{V} = \Big[p(x, x')\Big]_{x \in \mathcal{X} \setminus \mathcal{Y}, x' \in \mathcal{Y}} \qquad \mathbf{W} = \Big[p(x, x')\Big]_{x,x' \in \mathcal{X} \setminus \mathcal{Y}}$$

and let

$$\mathbf{P}_Y = \mathbf{T} + \mathbf{U}(\mathbf{I} - \mathbf{W})^{-1}\mathbf{V}.$$

Let $\{Y_t\}$ denote a Markov chain with state space $\mathcal{Y}$ and transition matrix $\mathbf{P}_Y$. Markov chain $\{Y_t\}$ is then irreducible and has stationary probabilities $\bar{p}_Y(y)$ that are given by

$$\bar{p}_Y(y) = \frac{\bar{p}(y)}{\sum_{y' \in \mathcal{Y}} \bar{p}(y')}$$

where $y \in \mathcal{Y}$.

Proof. See [32, Th. 6.1.1].[4]

Definition 11. Consider a set $\mathcal{X}$. For $x, x' \in \mathcal{X}$, the ordered pair (x, x') is called an *arrow* from x to x'. For $x_1, \ldots, x_N \in \mathcal{X}$, the sequence of arrows $((x_1, x_2), (x_2, x_3), \ldots, (x_{N-2}, x_{N-1}), (x_{N-1}, x_N))$ is called a *path* from x_1 to x_N. For $x \in \mathcal{X}$, a set of arrows E is called an *x-tree* on $\mathcal{X}$ if it satisfies the following conditions:

(1) E contains no arrow that starts at x.

(2) For each $x' \in \mathcal{X} \setminus \{x\}$, E contains exactly one arrow that starts at x'.

(3) For each $x' \in \mathcal{X} \setminus \{x\}$, E contains a path from x' to x (or, formulated more accurately, for each $x' \in \mathcal{X} \setminus \{x\}$, there exists a path from x' to x such that E contains all arrows of the path).

Lemma 5. *Let a Markov chain $\{X_t\}$ be irreducible. For $x \in \mathcal{X}$, let $\mathcal{E}(x)$ denote the set of all x-trees on $\mathcal{X}$. The stationary probabilities $\bar{p}(x)$ of the Markov chain are then given by*

$$\bar{p}(x) = \frac{\tilde{p}(x)}{\sum_{x' \in \mathcal{X}} \tilde{p}(x')}$$

where $x \in \mathcal{X}$ and

$$\tilde{p}(x) = \sum_{E \in \mathcal{E}(x)} \prod_{(x,x') \in E} p(x, x').$$

Proof. A proof is provided by Freidlin and Wentzell [22, Ch. 6, Lemma 3.1] (see also [19, Th. 4.2.1]).

[4]The terminology used in [32] differs from the terminology used in many other texts on Markov chains. In particular, an ergodic Markov chain in [32] corresponds to an irreducible Markov chain in this paper.

Using the above definitions and lemmas, we now prove the mathematical results presented in Section 2.

Proof of Lemma 1. Notice that

$$\Pr(W_{t+1} = w_{t+1} \,|\, W_t = w_t) = \Pr(W_{t+1} = w_{t+1} \,|\, W_t = w_t, \ldots, W_0 = w_0)$$

for all $t \in \{0, 1, \ldots\}$ and all $w_0, \ldots, w_{t+1} \in \mathcal{W}$. That is, the population in iteration $t + 1$ of a GA depends only on the population in iteration t. Given the population in iteration t, the population in iteration $t + 1$ is independent of the populations in iterations $0, \ldots, t - 1$. Notice further that

$$\Pr(W_{t+1} = w_{t+1} \,|\, W_t = w_t) = q(w_t, w_{t+1})$$

for all $t \in \{0, 1, \ldots\}$, all $w_t, w_{t+1} \in \mathcal{W}$, and some function $q : \mathcal{W}^2 \to [0, 1]$ that does not depend on t. That is, the probability of going from one population to some other population remains constant over time. (Recall that the crossover rate γ and the mutation rate ε are assumed to remain constant over time.) It now follows from Definitions 6 and 7 that $\{W_t\}$, where the index t takes values in $\{0, 1, \ldots\}$, is a Markov chain with state space $\mathcal{W}$ and transition probabilities $q(w, w')$. Since the mutation rate ε is assumed to be positive, any population can be turned into any other population in a single iteration of a GA. Hence, $q(w, w') > 0$ for all $w, w' \in \mathcal{W}$. Consequently, it follows from Definitions 9 and 10 that Markov chain $\{W_t\}$ is irreducible and aperiodic. Lemma 3 then implies that for each population $w \in \mathcal{W}$ there exists a stationary probability $\bar{q}(w)$ such that

$$\lim_{t \to \infty} \Pr(W_t = w \,|\, W_0 = w_0) = \bar{q}(w) \tag{18}$$

for all $w_0 \in \mathcal{W}$. We refer to a stationary probability $\bar{q}(w)$ as the long-run probability of population w. Finally, (2) is obtained from (18) by taking into account the time-homogeneity of Markov chain $\{W_t\}$. This completes the proof of Lemma 1. □

Proof of Theorem 1. As shown in the proof of Lemma 1, $\{W_t\}$, where the index t takes values in $\{0, 1, \ldots\}$, is an irreducible and aperiodic Markov chain with state space $\mathcal{W}$. Markov chain $\{W_t\}$ has stationary probabilities $\bar{q}(w)$, to which we refer as long-run probabilities. We now introduce some additional mathematical notation. Like in the proof of Lemma 1, the function $q : \mathcal{W}^2 \to [0, 1]$ denotes the transition probabilities of Markov chain $\{W_t\}$. For $w, w' \in \mathcal{W}$,

$q(w, w')$ is a polynomial in the mutation rate ε and can therefore be written as

$$q(w, w') = \sum_{l=0}^{\infty} \alpha(w, w', l)\varepsilon^l \tag{19}$$

where $\alpha(w, w', 0), \alpha(w, w', 1), \ldots$ denote the coefficients of the polynomial. $c(w, w')$ is defined as

$$c(w, w') = \min\{l \,|\, \alpha(w, w', l) \neq 0\}. \tag{20}$$

That is, $c(w, w')$ is defined as the rate at which $q(w, w')$ approaches zero as ε approaches zero. It follows from this definition that $c(w, w')$ equals the minimum number of mutations required to go from population w to population w' in a single iteration of a GA. $\alpha(w, w')$ is defined as

$$\alpha(w, w') = \alpha(w, w', c(w, w')). \tag{21}$$

For $w \in \mathcal{W}$, $\tilde{q}(w)$ is defined as

$$\tilde{q}(w) = \sum_{E \in \mathcal{E}(w)} \prod_{(w,w') \in E} q(w, w') \tag{22}$$

where $\mathcal{E}(w)$ denotes the set of all w-trees on $\mathcal{W}$. Since the transition probabilities $q(w, w')$ are polynomials in ε, $\tilde{q}(w)$ is a polynomial in ε too. $\tilde{q}(w)$ can therefore be written as

$$\tilde{q}(w) = \sum_{l=0}^{\infty} \tilde{\alpha}(w, l)\varepsilon^l \tag{23}$$

where $\tilde{\alpha}(w, 0), \tilde{\alpha}(w, 1), \ldots$ denote the coefficients of the polynomial. $\tilde{c}(w)$ is defined as

$$\tilde{c}(w) = \min\{l \,|\, \tilde{\alpha}(w, l) \neq 0\}. \tag{24}$$

That is, $\tilde{c}(w)$ is defined as the rate at which $\tilde{q}(w)$ approaches zero as ε approaches zero. $\tilde{\alpha}(w)$ is defined as

$$\tilde{\alpha}(w) = \tilde{\alpha}(w, \tilde{c}(w)). \tag{25}$$

Using the mathematical notation introduced above, we first prove part (i) of Theorem 1. It follows from (19), (20), and (22)–(24) that $\tilde{c}(w)$ can be written as

$$\tilde{c}(w) = \min_{E \in \mathcal{E}(w)} \sum_{(w,w') \in E} c(w, w'). \tag{26}$$

At least one mutation is required to go from a uniform population $u \in \mathcal{U}$ to any other population $w \in \mathcal{W} \setminus \{u\}$. Hence, $c(u, w) \geq 1$ for all $u \in \mathcal{U}$ and all $w \in \mathcal{W}$ such that $u \neq w$. Consequently,

it follows from (26) that $\tilde{c}(u) \geq \mu - 1$ for all $u \in \mathcal{U}$ and that $\tilde{c}(w) \geq \mu$ for all $w \in \mathcal{W} \setminus \mathcal{U}$. We now show that for each chromosome i it is possible to construct a $u(i)$-tree E on $\mathcal{W}$ that satisfies

$$\sum_{(w,w') \in E} c(w, w') = \mu - 1. \tag{27}$$

Consider an arbitrary chromosome i. Let the function $\rho : \mathcal{C} \to \mathcal{C}$ satisfy the following conditions:

(1) For each $j \neq i$, chromosome j is directly connected to chromosome $\rho(j)$.

(2) For each $j \neq i$, $\rho^N(j) = i$ for some positive integer N.

In condition (2), $\rho^N(j)$ is defined as

$$\rho^N(j) = \begin{cases} \rho(j), & \text{if } N = 1 \\ \rho\big(\rho^{N-1}(j)\big), & \text{otherwise.} \end{cases}$$

Because Theorem 1 assumes that each chromosome is connected to all other chromosomes, a function ρ that satisfies the above two conditions is guaranteed to exist. In order to construct a $u(i)$-tree E on $\mathcal{W}$ that satisfies (27), we start with an empty set of arrows E. For each $j \neq i$, we then add an arrow to E that starts at $u(j)$ and ends at $v(j, \rho(j), n-1)$. It follows from condition (1) that one mutation is required to go from $u(j)$ to $v(j, \rho(j), n-1)$ in a single iteration of a GA. Hence, $c(u(j), v(j, \rho(j), n-1)) = 1$. Next, for each $j \neq i$, we add a path to E that starts at $v(j, \rho(j), n-1)$ and ends at $u(\rho(j))$. Each path that we add to E must contain no cycles, that is, it must contain no two arrows (w_1, w_1') and (w_2, w_2') such that either $w_1 = w_2$ or $w_1' = w_2'$. In addition, each path must only contain arrows (w, w') for which $c(w, w') = 0$. Condition (1) guarantees the existence of paths that satisfy the latter requirement. Due to condition (2), for each $u \in \mathcal{U} \setminus \{u(i)\}$, E now contains a path from u to $u(i)$. Finally, for each $w \in \mathcal{W} \setminus \mathcal{U}$, if E does not yet contain an arrow that starts at w, we add such an arrow to E. We choose the arrows that we add to E in such a way that, after adding the arrows, E contains, for each $w \in \mathcal{W} \setminus \mathcal{U}$, a path from w to some $u \in \mathcal{U}$ (which implies that E contains a path from w to $u(i)$). In addition, we only choose arrows (w, w') for which $c(w, w') = 0$. We can choose the arrows in this way because Theorem 1 assumes that all non-uniform populations are almost uniform. Using Definition 11, it can be seen that the set of arrows E constructed as discussed above is a $u(i)$-tree on $\mathcal{W}$. Moreover, E satisfies (27). We have therefore shown that

for each chromosome i a $u(i)$-tree E on $\mathcal{W}$ that satisfies (27) can be constructed. Consequently, it follows from (26) that $\tilde{c}(u) \leq \mu - 1$ for all $u \in \mathcal{U}$. Since it has been shown above that $\tilde{c}(u) \geq \mu - 1$ for all $u \in \mathcal{U}$, this implies that $\tilde{c}(u) = \mu - 1$ for all $u \in \mathcal{U}$. It has also been shown above that $\tilde{c}(w) \geq \mu$ for all $w \in \mathcal{W} \setminus \mathcal{U}$. Hence, as the mutation rate ε approaches zero, $\tilde{q}(w)$ approaches zero faster for $w \in \mathcal{W} \setminus \mathcal{U}$ than for $w \in \mathcal{U}$. It then follows from Lemma 5 that for all non-uniform populations $w \in \mathcal{W} \setminus \mathcal{U}$ the long-run probability $\bar{q}(w)$ approaches zero as ε approaches zero. In other words, the long-run limit probability $\hat{q}(w)$ equals zero for all non-uniform populations $w \in \mathcal{W} \setminus \mathcal{U}$. This completes the proof of part (i) of Theorem 1.

We now prove part (ii) of Theorem 1. It has been shown above that $\tilde{c}(u) = \mu - 1$ for all $u \in \mathcal{U}$. Consequently, as the mutation rate ε approaches zero, $\tilde{q}(u)$ approaches zero equally fast for all $u \in \mathcal{U}$. Using Lemma 5, it can therefore be seen that the long-run limit probability $\hat{q}(u)$ of a uniform population $u \in \mathcal{U}$ is given by

$$\hat{q}(u) = \lim_{\varepsilon \to 0} \bar{q}(u) = \frac{\tilde{\alpha}(u)}{\sum_{u' \in \mathcal{U}} \tilde{\alpha}(u')}. \tag{28}$$

For $u \in \mathcal{U}$, let $\widetilde{\mathcal{E}}(u)$ be defined as

$$\widetilde{\mathcal{E}}(u) = \Big\{ E \in \mathcal{E}(u) \,\Big|\, \textstyle\sum_{(w,w') \in E} c(w,w') = \mu - 1 \Big\}. \tag{29}$$

It then follows from (19)–(25) that $\tilde{\alpha}(u)$ can be written as

$$\tilde{\alpha}(u) = \sum_{E \in \widetilde{\mathcal{E}}(u)} \prod_{(w,w') \in E} \alpha(w,w'). \tag{30}$$

Consider an arbitrary uniform population $u \in \mathcal{U}$ and an arbitrary u-tree E on $\mathcal{W}$, where $E \in \widetilde{\mathcal{E}}(u)$. Let E_1 and E_2 denote sets of arrows that are given by

$$E_1 = \{(w,w') \in E \,|\, w \in \mathcal{V}\}$$
$$E_2 = E \setminus E_1.$$

It is immediately clear that E_1 satisfies the following conditions:

(A1) E_1 contains no arrow that starts at u or at some $w \in \mathcal{W} \setminus \mathcal{V}$.

(A2) For each $v \in \mathcal{V} \setminus \{u\}$, E_1 contains exactly one arrow that starts at v.

Notice that $c(u',w) \geq 1$ for all $u' \in \mathcal{U}$ and all $w \in \mathcal{W}$ such that $u' \neq w$. Notice further that, due to (29), $\sum_{(w,w') \in E_1} c(w,w') \leq \mu - 1$. These observations imply that, for each $(w,w') \in E_1$, $c(w,w') = 1$ if $w \in \mathcal{U}$ and $c(w,w') = 0$ otherwise. They also imply that E_1 satisfies the following condition:

(A3) $\displaystyle\sum_{(w,w')\in E_1} c(w,w') = \mu - 1.$

It is easy to see that $c(v,w) \geq 1$ for all $v \in \mathcal{V}$ and all $w \in \mathcal{W} \setminus \mathcal{V}$ and that $c(u',w) \geq 2$ for all $u' \in \mathcal{U}$ and all $w \in \mathcal{W} \setminus \mathcal{V}$. Consequently, E_1 contains no arrows that end at some $w \in \mathcal{W} \setminus \mathcal{V}$. This implies the following condition on E_1:

(A4) For each $v \in \mathcal{V} \setminus \{u\}$, E_1 contains a path from v to u.

It is immediately clear that E_2 satisfies the following conditions:

(B1) E_2 contains no arrow that starts at some $v \in \mathcal{V}$.

(B2) For each $w \in \mathcal{W} \setminus \mathcal{V}$, E_2 contains exactly one arrow that starts at w.

(B3) For each $w \in \mathcal{W} \setminus \mathcal{V}$, E_2 contains a path from w to some $v \in \mathcal{V}$.

Furthermore, taking into account that E_1 satisfies condition (A3), (29) implies that E_2 satisfies the following condition:

(B4) $\displaystyle\sum_{(w,w')\in E_2} c(w,w') = 0.$

For $u \in \mathcal{U}$, let $\widetilde{\mathcal{E}}_1(u)$ denote a set that contains all sets of arrows E_1 satisfying conditions (A1)–(A4). Let $\widetilde{\mathcal{E}}_2$ denote a set that contains all sets of arrows E_2 satisfying conditions (B1)–(B4). Notice that $\widetilde{\mathcal{E}}_2$ does not depend on u. Clearly, for each $E \in \widetilde{\mathcal{E}}(u)$, there exist an $E_1 \in \widetilde{\mathcal{E}}_1(u)$ and an $E_2 \in \widetilde{\mathcal{E}}_2$ such that $E = E_1 \cup E_2$. Conversely, it can be seen that for each $E_1 \in \widetilde{\mathcal{E}}_1(u)$ and each $E_2 \in \widetilde{\mathcal{E}}_2$ there exists an $E \in \widetilde{\mathcal{E}}(u)$ such that $E = E_1 \cup E_2$. Hence,

$$\widetilde{\mathcal{E}}(u) = \Big\{ E_1 \cup E_2 \,\Big|\, E_1 \in \widetilde{\mathcal{E}}_1(u), E_2 \in \widetilde{\mathcal{E}}_2 \Big\}.$$

Equation (30) can now be written as

$$\tilde{\alpha}(u) = \Big(\textstyle\sum_{E_1\in\widetilde{\mathcal{E}}_1(u)} \prod_{(w,w')\in E_1} \alpha(w,w')\Big)\Big(\sum_{E_2\in\widetilde{\mathcal{E}}_2} \prod_{(w,w')\in E_2} \alpha(w,w')\Big).$$

Consequently, it follows from (28) that

$$\hat{q}(u) = \lim_{\varepsilon\to 0} \bar{q}(u) = \frac{\sum_{E_1\in\widetilde{\mathcal{E}}_1(u)} \prod_{(w,w')\in E_1} \alpha(w,w')}{\sum_{u'\in\mathcal{U}} \sum_{E_1\in\widetilde{\mathcal{E}}_1(u')} \prod_{(w,w')\in E_1} \alpha(w,w')}. \tag{31}$$

Based on (31), the following observations can be made:

(1) For $w, w' \in \mathcal{W}$ such that $w \neq w'$ and such that there exists an $E_1 \in \bigcup_{u' \in \mathcal{U}} \widetilde{\mathcal{E}}_1(u')$ that contains an arrow (w, w'), $\lim_{\varepsilon \to 0} \bar{q}(u)$ depends on the term of lowest degree in the transition probability $q(w, w')$ and does not depend on other terms in $q(w, w')$.

(2) For $w, w' \in \mathcal{W}$ such that $w \neq w'$ and such that there does not exist an $E_1 \in \bigcup_{u' \in \mathcal{U}} \widetilde{\mathcal{E}}_1(u')$ that contains an arrow (w, w'), $\lim_{\varepsilon \to 0} \bar{q}(u)$ does not depend on any of the terms in the transition probability $q(w, w')$.

Let $\{V_t\}$, where the index t takes values in $\{0, 1, \ldots\}$, denote a Markov chain with state space $\mathcal{V}$, transition probabilities $r(v, v')$, and stationary probabilities $\bar{r}(v)$, where $v, v' \in \mathcal{V}$. For $v \neq v'$, let

$$r(v, v') = \begin{cases} \alpha(v, v')\varepsilon, & \text{if } v \in \mathcal{U} \text{ and } c(v, v') = 1 \\ \alpha(v, v'), & \text{if } v \notin \mathcal{U} \text{ and } c(v, v') = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{32}$$

Furthermore, let $r(v, v) = 1 - \sum_{v' \in \mathcal{V} \setminus \{v\}} r(v, v')$. Clearly, Markov chain $\{V_t\}$ is irreducible. Taking into account the two observations made above, it can be seen that $\lim_{\varepsilon \to 0} \bar{r}(v) = \lim_{\varepsilon \to 0} \bar{q}(v)$ for all $v \in \mathcal{V}$. That is, in the limit as ε approaches zero, corresponding states of Markov chains $\{V_t\}$ and $\{W_t\}$ have the same stationary probability. It follows from this that $\lim_{\varepsilon \to 0} \bar{r}(v) = \hat{q}(v)$ for all $v \in \mathcal{V}$.

The following observations can be made:

(1) For $v \in \mathcal{U}$ and $v' \in \mathcal{V}$, $c(v, v') = 1$ if and only if $v = u(i)$ and $v' = v(i, j, n-1)$ for some i and some j such that $\delta(i, j) = 1$.

(2) For $v \in \mathcal{U}$ and $v' \in \mathcal{V}$ such that $c(v, v') = 1$, $q(v, v')$ equals the probability that the mutation operator inverts one specific bit in the binary encoding of an arbitrarily chosen chromosome and that it does not invert any other bits in the binary encoding of the chosen chromosome or of any other chromosome in the population. This probability does not depend on v or v'. Consequently, for all $v_1, v_2 \in \mathcal{U}$ and all $v_1', v_2' \in \mathcal{V}$ such that $c(v_1, v_1') = c(v_2, v_2') = 1$, $q(v_1, v_1') = q(v_2, v_2')$ and hence $\alpha(v_1, v_1') = \alpha(v_2, v_2')$.

(3) For $v \in \mathcal{V} \setminus \mathcal{U}$ and $v' \in \mathcal{V}$, $c(v, v') = 0$ only if $v = v(i, j, \lambda)$ and $v' = v(i, j, \lambda')$ for some i and some j such that $\delta(i, j) = 1$ and for some $\lambda \in \{1, \ldots, n-1\}$ and some $\lambda' \in \{0, \ldots, n\}$.

(4) For $v \in \mathcal{V} \setminus \mathcal{U}$ and $v' \in \mathcal{V}$ such that $c(v, v') = 0$, $\alpha(v, v') = \pi(i, j, \lambda, \lambda')$, where i, j, λ, and λ' satisfy $v = v(i, j, \lambda)$ and $v' = v(i, j, \lambda')$ and where $\pi(i, j, \lambda, \lambda')$ is defined in (1).

Let $\alpha = \alpha(v, v')$ for all $v \in \mathcal{U}$ and all $v' \in \mathcal{V}$ such that $c(v, v') = 1$. Using (32), it follows from the first two observations made above that $r(v, v') = \alpha\varepsilon$ if $v = u(i)$ and $v' = v(i, j, n-1)$ for some i and some j such that $\delta(i, j) = 1$. It also follows that $r(v, v') = 1 - m\alpha\varepsilon$ if $v = v' \in \mathcal{U}$. Furthermore, taking into account the last two observations made above, it can be seen from (32) that $r(v, v') = \pi(i, j, \lambda, \lambda')$ if $v = v(i, j, \lambda)$ and $v' = v(i, j, \lambda')$ for some i and some j such that $\delta(i, j) = 1$ and for some $\lambda \in \{1, \ldots, n-1\}$ and some $\lambda' \in \{0, \ldots, n\}$. Finally, (32) implies that $r(v, v') = 0$ if none of the above conditions is satisfied. Let the vector $\tilde{\mathbf{v}} = \begin{bmatrix} \tilde{v}_1 & \cdots & \tilde{v}_\xi \end{bmatrix}$ be given by

$$\tilde{\mathbf{v}}^{\mathrm{T}} = \begin{bmatrix} v(g_1, h_1, 1) \\ \vdots \\ v(g_1, h_1, n-1) \\ v(g_2, h_2, 1) \\ \vdots \\ \vdots \\ v(g_{\nu-1}, h_{\nu-1}, n-1) \\ v(g_\nu, h_\nu, 1) \\ \vdots \\ v(g_\nu, h_\nu, n-1) \end{bmatrix}$$

where $\mathbf{g} = [g_k]$ and $\mathbf{h} = [h_k]$ are defined in Section 2. Notice that $\tilde{\mathbf{v}}$ contains each population in $\mathcal{V} \setminus \mathcal{U}$ exactly once. It can be seen that

$$(1 - m\alpha\varepsilon)\mathbf{I} = \begin{bmatrix} r(u(0), u(0)) & \cdots & r(u(0), u(\mu-1)) \\ \vdots & \ddots & \vdots \\ r(u(\mu-1), u(0)) & \cdots & r(u(\mu-1), u(\mu-1)) \end{bmatrix} \tag{33}$$

$$\alpha\varepsilon\mathbf{A} = \begin{bmatrix} r(u(0), \tilde{v}_1) & \cdots & r(u(0), \tilde{v}_\xi) \\ \vdots & \ddots & \vdots \\ r(u(\mu-1), \tilde{v}_1) & \cdots & r(u(\mu-1), \tilde{v}_\xi) \end{bmatrix} \tag{34}$$

$$\mathbf{B} = \begin{bmatrix} r(\tilde{v}_1, u(0)) & \cdots & r(\tilde{v}_1, u(\mu-1)) \\ \vdots & \ddots & \vdots \\ r(\tilde{v}_\xi, u(0)) & \cdots & r(\tilde{v}_\xi, u(\mu-1)) \end{bmatrix} \tag{35}$$

$$\mathbf{C} = \begin{bmatrix} r(\tilde{v}_1, \tilde{v}_1) & \cdots & r(\tilde{v}_1, \tilde{v}_\xi) \\ \vdots & \ddots & \vdots \\ r(\tilde{v}_\xi, \tilde{v}_1) & \cdots & r(\tilde{v}_\xi, \tilde{v}_\xi) \end{bmatrix} \tag{36}$$

where $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are defined in (3), (6), and (9). Let $\mathbf{S}$ denote a $\mu \times \mu$ matrix that is obtained from the matrices in (33)–(36) and that is given by

$$\mathbf{S} = (1 - m\alpha\varepsilon)\mathbf{I} + \alpha\varepsilon\mathbf{A}(\mathbf{I} - \mathbf{C})^{-1}\mathbf{B}. \tag{37}$$

This can be written more simply as

$$\mathbf{S} = \mathbf{I} + \alpha\varepsilon\mathbf{D}$$

where $\mathbf{D}$ is defined in (12). Let $\{U_t\}$, where the index t takes values in $\{0, 1, \ldots\}$, denote a Markov chain with state space $\mathcal{U}$ and transition matrix $\mathbf{S}$. Using (33)–(37), it follows from Lemma 4 that Markov chain $\{U_t\}$ is irreducible and has stationary probabilities $\bar{s}(u)$ that are given by

$$\bar{s}(u) = \frac{\bar{r}(u)}{\sum_{u' \in \mathcal{U}} \bar{r}(u')} \tag{38}$$

where $u \in \mathcal{U}$. Definition 8 states that the stationary distribution $\bar{\mathbf{s}} = \begin{bmatrix} \bar{s}(u(0)) & \cdots & \bar{s}(u(\mu - 1)) \end{bmatrix}$ of Markov chain $\{U_t\}$ satisfies

$$\bar{\mathbf{s}}\mathbf{S} = \bar{\mathbf{s}} \tag{39}$$

$$\bar{\mathbf{s}}\mathbf{1} = 1. \tag{40}$$

Lemma 2 implies that this linear system has a unique solution. The equality in (39) can be written as

$$\bar{\mathbf{s}}(\mathbf{S} - \mathbf{I}) = \alpha\varepsilon\bar{\mathbf{s}}\mathbf{D} = \mathbf{0}.$$

Since $\alpha > 0$ and $\varepsilon > 0$, this can be simplified to

$$\bar{\mathbf{s}}\mathbf{D} = \mathbf{0}. \tag{41}$$

Notice that $\mathbf{D}$ does not depend on ε. $\bar{\mathbf{s}}$ therefore does not depend on ε either. Recall further that $\lim_{\varepsilon \to 0} \bar{r}(v) = \hat{q}(v)$ for all $v \in \mathcal{V}$ and that $\hat{q}(w) = 0$ for all $w \in \mathcal{W} \setminus \mathcal{U}$. Using (38), it now follows that

$$\bar{s}(u) = \lim_{\varepsilon \to 0} \bar{s}(u) = \lim_{\varepsilon \to 0} \frac{\bar{r}(u)}{\sum_{u' \in \mathcal{U}} \bar{r}(u')} = \frac{\hat{q}(u)}{\sum_{u' \in \mathcal{U}} \hat{q}(u')} = \hat{q}(u)$$

for all $u \in \mathcal{U}$. Hence, the stationary distribution $\bar{\mathbf{s}}$ of Markov chain $\{U_t\}$ equals the long-run limit distribution $\hat{\mathbf{q}}$. Consequently, (40) and (41) imply that $\hat{\mathbf{q}}$ satisfies (13) and (14). It also follows that the linear system given by (13) and (14) has a unique solution. This completes the proof of part (ii) of Theorem 1. □

Acknowledgment

The authors would like to thank Uzay Kaymak and Rommert Dekker for their comments on earlier versions of this paper.

References

[1] F. Alkemade, J. La Poutré, and H. Amman. Robust evolutionary algorithm design for socio-economic simulation. *Computational Economics*, 28(4):355–370, 2006.

[2] F. Alkemade, J. La Poutré, and H. Amman. On social learning and robust evolutionary algorithm design in the Cournot oligopoly game. *Computational Intelligence*, 23(2):162–175, 2007.

[3] F. Alkemade, J. La Poutré, and H. Amman. Robust evolutionary algorithm design for socio-economic simulation: A correction. *Computational Economics*, 33(1):99–101, 2009.

[4] F. Alkemade, D. Van Bragt, and J. La Poutré. Stabilization of tag-mediated interaction by sexual reproduction in an evolutionary agent system. *Information Sciences*, 170(1):101–119, 2005.

[5] J. Andreoni and J. Miller. Auctions with artificial adaptive agents. *Games and Economic Behavior*, 10(1):39–64, 1995.

[6] J. Arifovic. Genetic algorithm learning and the cobweb model. *Journal of Economic Dynamics and Control*, 18(1):3–28, 1994.

[7] J. Arifovic. The behavior of the exchange rate in the genetic algorithm and experimental economies. *Journal of Political Economy*, 104(3):510–541, 1996.

[8] D. Ashlock and E.-Y. Kim. Fingerprinting: Visualization and automatic analysis of prisoner's dilemma strategies. *IEEE Transactions on Evolutionary Computation*, 12(5):647–659, 2008.

[9] D. Ashlock, E.-Y. Kim, and N. Leahy. Understanding representational sensitivity in the iterated prisoner's dilemma with fingerprints. *IEEE Transactions on Systems, Man, and Cybernetics C*, 36(4):464–475, 2006.

[10] D. Ashlock, M. Smucker, E. Stanley, and L. Tesfatsion. Preferential partner selection in an evolutionary study of prisoner's dilemma. *Biosystems*, 37(1–2):99–125, 1996.

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

[12] R. Axelrod. The evolution of strategies in the iterated prisoner's dilemma. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 32–41. Morgan Kaufmann, 1987.

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

[14] R. Barrett et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Second edition, 2008. Available online at `http://www.netlib.org/linalg/html_templates/Templates.html`.

[15] T. Brenner. Agent learning representation: Advice on modelling economic learning. In L. Tesfatsion and K. Judd, editors, *Handbook of Computational Economics, Volume 2*, pages 895–947. Elsevier, 2006.

[16] S. Chong and X. Yao. Behavioral diversity, choices and noise in the iterated prisoner's dilemma. *IEEE Transactions on Evolutionary Computation*, 9(6):540–551, 2005.

[17] S. Chong and X. Yao. Multiple choices and reputation in multiagent interactions. *IEEE Transactions on Evolutionary Computation*, 11(6):689–711, 2007.

[18] P. Crowley, L. Provencher, S. Sloane, L. Dugatkin, B. Spohn, L. Rogers, and M. Alfieri. Evolving cooperation: The role of individual recognition. *Biosystems*, 37(1–2):49–66, 1996.

[19] H. Dawid. *Adaptive Learning by Genetic Algorithms: Analytical Results and Applications to Economic Models*. Number 441 in Lecture Notes in Economics and Mathematical Systems. Springer, 1996.

[20] D. Fogel. Evolving behaviors in the iterated prisoner's dilemma. *Evolutionary Computation*, 1(1):77–97, 1993.

[21] D. Foster and P. Young. Stochastic evolutionary game dynamics. *Theoretical Population Biology*, 38(2):219–232, 1990.

[22] M. Freidlin and A. Wentzell. *Random Perturbations of Dynamical Systems*. Springer, second edition, 1998.

[23] D. Fudenberg and D. Levine. *The Theory of Learning in Games*. MIT Press, 1998.

[24] M. Gen and R. Cheng. *Genetic Algorithms & Engineering Optimization*. John Wiley & Sons, 2000.

[25] C. Georges. Learning with misspecification in an artificial currency market. *Journal of Economic Behavior and Organization*, 60(1):70–84, 2006.

[26] H. Gintis. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction*. Princeton University Press, 2000.

[27] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[28] E. Haruvy, A. Roth, and M. Utku Ünver. The dynamics of law clerk matching: An experimental and computational investigation of proposals for reform of the market. *Journal of Economic Dynamics and Control*, 30(3):457–486, 2006.

[29] J. Holland and J. Miller. Artificial adaptive agents in economic theory. *American Economic Review (Papers and Proceedings)*, 81(2):365–370, 1991.

[30] H. Ishibuchi and N. Namikawa. Evolution of iterated prisoner's dilemma game strategies in structured demes under random pairing in game playing. *IEEE Transactions on Evolutionary Computation*, 9(6):552–561, 2005.

[31] M. Kandori, G. Mailath, and R. Rob. Learning, mutation, and long run equilibria in games. *Econometrica*, 61:29–56, 1993.

[32] J. Kemeny and J. Snell. *Finite Markov Chains*. D. Van Nostrand Company, 1960.

[33] T. Lux and S. Schornstein. Genetic learning as an explanation of stylized facts of foreign exchange markets. *Journal of Mathematical Economics*, 41(1–2):169–196, 2005.

[34] R. Marks. Breeding hybrid strategies: Optimal behaviour for oligopolists. *Journal of Evolutionary Economics*, 2(1):17–38, 1992.

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

[36] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, third edition, 1996.

[37] J. Miller. A genetic model of adaptive economic behavior. Working paper, University of Michigan, 1986.

[38] J. Miller. The coevolution of automata in the repeated prisoner's dilemma. *Journal of Economic Behavior and Organization*, 29(1):87–112, 1996.

[39] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[40] S. Mittal and K. Deb. Optimal strategies of the iterated prisoner's dilemma problem for multiple conflicting objectives. In *Proceedings of the 2006 IEEE Symposium on Computational Intelligence and Games*, pages 197–204, 2006.

[41] H. Mühlenbein. Darwin's continent cycle theory and its simulation by the prisoner's dilemma. *Complex Systems*, 5(5):459–478, 1991.

[42] A. Nix and M. Vose. Modeling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, 5(1):79–88, 1992.

[43] G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.

[44] G. Rudolph. Finite Markov chain results in evolutionary computation: A tour d'horizon. *Fundamenta Informaticae*, 35(1–4):67–89, 1998.

[45] W. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[46] L. Tesfatsion. Agent-based computational economics: A constructive approach to economic theory. In L. Tesfatsion and K. Judd, editors, *Handbook of Computational Economics, Volume 2*, pages 831–880. Elsevier, 2006.

[47] X. Thibert-Plante and P. Charbonneau. Crossover and evolutionary stability in the prisoner's dilemma. *Evolutionary Computation*, 15(3):321–344, 2007.

[48] H. Tijms. *Stochastic Models: An Algorithmic Approach*. John Wiley & Sons, 1994.

[49] H. Tijms. *A First Course in Stochastic Models*. John Wiley & Sons, 2003.

[50] D. Van Bragt, C. Van Kemenade, and J. La Poutré. The influence of evolutionary selection schemes on the iterated prisoner's dilemma. *Computational Economics*, 17(2–3):253–263, 2001.

[51] F. Vega-Redondo. *Evolution, Games, and Economic Behaviour*. Oxford University Press, 1996.

[52] F. Vega-Redondo. The evolution of Walrasian behavior. *Econometrica*, 65:375–384, 1997.

[53] M. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, 1999.

[54] N. Vriend. An illustration of the essential difference between individual and social learning, and its consequences for computational analyses. *Journal of Economic Dynamics and Control*, 24(1):1–19, 2000.

[55] L. Waltman and N. Van Eck. Robust evolutionary algorithm design for socio-economic simulation: Some comments. *Computational Economics*, 33(1):103–105, 2009.

[56] J. Weibull. *Evolutionary Game Theory*. MIT Press, 1995.

[57] X. Yao and P. Darwen. An experimental study of N-person iterated prisoner's dilemma games. *Informatica*, 18(4):435–450, 1994.

[58] H. Young. The evolution of conventions. *Econometrica*, 61:57–84, 1993.

Publications in the Report Series Research* in Management

ERIM Research Program: "Business Processes, Logistics and Information Systems"

2009

How to Normalize Co-Occurrence Data? An Analysis of Some Well-Known Similarity Measures
Nees Jan van Eck and Ludo Waltman
ERS-2009-001-LIS
http://hdl.handle.net/1765/14528

Spare Parts Logistics and Installed Base Information
Muhammad N. Jalil, Rob A. Zuidwijk, Moritz Fleischmann, and Jo A.E.E. van Nunen
ERS-2009-002-LIS
http://hdl.handle.net/1765/14529

Open Location Management in Automated Warehousing Systems
Yugang YU and René B.M. de Koster
ERS-2009-004-LIS
http://hdl.handle.net/1765/14615

VOSviewer: A Computer Program for Bibliometric Mapping
Nees Jan van Eck and Ludo Waltman
ERS-2009-005-LIS
http://hdl.handle.net/1765/14841

Nash Game Model for Optimizing Market Strategies, Configuration of Platform Products in a Vendor Managed Inventory (VMI) Supply Chain for a Product Family
Yugang Yu and George Q. Huang
ERS-2009-009-LIS
http://hdl.handle.net/1765/15029

A Mathematical Analysis of the Long-run Behavior of Genetic Algorithms for Social Modeling
Ludo Waltman and Nees Jan van Eck
ERS-2009-011-LIS
http://hdl.handle.net/1765/15181

A Taxonomy of Bibliometric Performance Indicators Based on the Property of Consistency
Ludo Waltman and Nees Jan van Eck
ERS-2009-014-LIS
http://hdl.handle.net/1765/15182

A Stochastic Dynamic Programming Approach to Revenue Management in a Make-to-Stock Production System
Rainer Quante, Moritz Fleischmann, and Herbert Meyr
ERS-2009-015-LIS
http://hdl.handle.net/1765/15183

Some Comments on Egghe's Derivation of the Impact Factor Distribution
Ludo Waltman and Nees Jan van Eck
ERS-2009-016-LIS
http://hdl.handle.net/1765/15184

---

* A complete overview of the ERIM Report Series Research in Management:
https://ep.eur.nl/handle/1765/1

ERIM Research Programs:
LIS Business Processes, Logistics and Information Systems
ORG Organizing for Performance
MKT Marketing
F&A Finance and Accounting
STR Strategy and Entrepreneurship