Finite Population Models of Co-evolution and Their Application to Haploidy versus Diploidy

Anthony M.L. Liekens, Huub M.M. ten Eikelder, and Peter A.J. Hilbers

Department of Biomedical Engineering
Technische Universiteit Eindhoven
P.O. Box 513, 5600MB Eindhoven, The Netherlands
{a.m.l.liekens, h.m.m.t.eikelder, p.a.j.hilbers}@tue.nl

Abstract. In order to study genetic algorithms in co-evolutionary environments, we construct a Markov model of co-evolution of populations with fixed, finite population sizes. In this combined Markov model, the behavior toward the limit can be utilized to study the relative performance of the algorithms. As an application of the model, we perform an analysis of the relative performance of haploid versus diploid genetic algorithms in the co-evolutionary setup, under several parameter settings. Because of the use of Markov chains, this paper provides exact stochastic results on the expected performance of haploid and diploid algorithms in the proposed co-evolutionary model.

1 Introduction

Co-evolution of Genetic Algorithms (GA) denotes the simultaneous evolution of two or more GAs with interdependent or coupled fitness functions. In competitive co-evolution, just like competition in nature, individuals of both algorithms compete with each other to gather fitness. In cooperative co-evolution, individuals have to cooperate to achieve higher fitness. These interactions have previously been modeled in Evolutionary Game Theory (EGT), using replicator dynamics and infinite populations. Similar models have, for example, been used to study equilibriums [2] and comparisons of selection methods [1]. Simulations of competitive co-evolution have previously been used to evolve solutions and strategies for small two-player games, i.e., in [3,4], sorting networks [5], or competitive robotics [6].

In this paper, we provide the construction of a Markov model of co-evolution of two GAs with finite population sizes. After this construction we calculate the relative performances in such a setup, in which a haploid and diploid GA co-evolve with each other.

Commonly, GAs are based on the haploid model of reproduction. In this model, an individual is assumed to carry a single genotype to encode for its phenotype. When two parents are selected for reproduction, recombination of these two genotypes takes place to construct a child for the next generation.

Most higher order species in nature, however, have the characteristic of carrying two sets of alleles that both can encode for the individual's phenotype.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 344–355, 2003.

For each of the genes, two (possibly different) alleles are thus present. A dominance relation is defined on each pair of alleles. In a heterozygous gene, i.e., in a gene with 2 different alleles, this dominance relation defines which allele is expressed. A dominance relation can be pure, such that either one of the alleles is always expressed in heterozygous individuals, or it can be partial, such that the result of phenotypic expression is a probability distribution over the alleles. When two diploid parents are selected to reproduce, they produce haploid gamete cells through meiosis, in which each parent's genes are recombined. The haploid gametes are then merged, or fertilized, to form a new diploid child.

In dynamic environments, diploid GAs are hypothesized to perform better than haploid algorithms, since they can build up an implicit long time memory of previously encountered solutions in the recessive parts of the populations' allele pool. These alleles are kept safe from harmful selection. Under the assumption that co-evolution mimics a dynamic environment, we will test this hypothesis with a small problem in this paper, using co-evolution as a special form of dynamic optimization. The Markov model approach yields exact stochastic expectations of performance of haploid and diploid algorithms. Previous accounts of research on the use of diploidy for dynamic optimization, and results of its performance as compared with haploid algorithms, can be found in [5,7,8,9,10]. The methods used in these papers differ from our approach in the fact that we consider exact probability distributions whereas others perform simulation experiments or equilibrium analyses of infinite models. The stochastic method of Markov models, as used in this paper, allows us to provide exact stochastic results and performance expectations, instead of empirical data which is, as we will show later, subject to a large standard deviation. A similar model to the model presented in this paper, discussing stochastic models for dynamic optimization problems, is discussed in [11].

In this study, haploid and diploid populations face one another in co-evolution, which creates a simulation of a comparable situation in the history of life on Earth: The first diploid organisms to appear on Earth had to face haploid life forms in a competition for resources. The dynamics of the co-evolutionary competitive games played by these prehistoric cells are similar to the models presented in this paper. Correct interpretation of the results can give insights whether the earliest diploid life forms were able to compete with haploid life forms.

In this paper, co-evolution, of two competing populations and their governing GAs, is used as a "test bed" to test two algorithms' relative performance in dynamic environments. Indeed, since the fitness of an individual in one of the co-evolving populations is based on the configuration of the opponent population, the fitness landscapes of both populations constantly change, thereby simulating dynamic environments through both populations' interdependent fitness functions. Note that the results can only be used to discuss the algorithms' *relative* performance since the dynamics of one algorithm is explicitly determined by the other algorithm.

2 Models and Methods

In this section, we construct a finite population Markov model of co-evolution. Two finite population Markov chains of simple genetic algorithms, based on the simple GA as described by [12,13], are intertwined through interdependent fitness functions. A discussion of the resulting Markov chain's behavior toward the limit and the interpretation of the limit behavior is also provided.

2.1 Haploid and Diploid Reproduction Schemes

The following constructions are based on the definition of haploid and diploid simple genetic algorithms with finite population sizes as described in [13].

Haploid Reproduction. Let Ω_H be the space of binary bit strings with length l. The bit string serves as a genotype with l loci, that each can hold the alleles 0 or 1. Ω_H serves as the search space for the Haploid Simple Genetic Algorithm (HSGA). Let P_H be a haploid population, $P_H = \{x_0, x_1, \ldots, x_{r_H-1}\}$, a multi set with $x_i \in \Omega_H$ for $0 \leq i < r_H$, and $r_H = |P_H|$ the population size. Let π_H denote the set of all possible populations P_H of size r_H.

Let $f_H : \Omega_H \to \mathbb{R}^+$ denote the fitness function. Let $\varsigma_{f_H} : \pi_H \to \Omega_H$ represent stochastic selection, proportional to fitness function f_H. Crossover is a genetic operator that takes two parent individuals, and results in a new child individual that shares properties of these parents. Mutation slightly changes the genotype of an individual. Crossover and mutation are represented by the stochastic functions $\chi : \Omega_H \times \Omega_H \to \Omega_H$ and $\mu : \Omega_H \to \Omega_H$ respectively.

In a HSGA, a new generation of individuals is created through sexual reproduction of selected parents from the current population. The probability that a haploid individual $i \in \Omega_H$ is generated from a population P_H can be written according to this process as

$$\begin{aligned} &\mathtt{Pr}\left[i \text{ is generated from } P_H\right] = \\ &\mathtt{Pr}\left[\mu\left(\chi\left(\varsigma_{f_H}\left(P_H\right), \varsigma_{f_H}\left(P_H\right)\right)\right) = i\right] \end{aligned} \tag{1}$$

where it has been shown in [13] that the order of mutation and crossover may be interchanged in equation (1).

Diploid Reproduction. In the Diploid Simple Genetic Algorithm (DSGA), an individual consists of two haploid genomes. An individual of the diploid population is represented by a multi set of two instances of Ω_H, e.g. $\{i, j\}$ with $i, j \in \Omega_H$. The set of all possible diploid instances is denoted by Ω_D, the search space of the DSGA. A diploid population P_D with population size r_D is defined over Ω_D, similar to the definition of a haploid population. Let π_D denote the set of possible populations.

Haploid selection, mutation and crossover are reused in the diploid algorithm. Two more specific genetic operators must be defined. $\delta : \Omega_D \to \Omega_H$

is the dominance operator. A fitness function f_H defined for the haploid algorithm, can be reused in a fitness function f_D for the diploid algorithm with $f_D(\{i,j\}) = f_H(\delta(\{i,j\}))$ for any $\{i,j\}$ in Ω_D. Another diploid-specific operator is fertilization, which merges two gametes (members of Ω_H) into one diploid individual: $\phi : \Omega_H \times \Omega_H \to \Omega_D$. Throughout this paper we will assume that $\phi(i,j) = \{i,j\}$ for all i,j in Ω_H. Diploid reproduction can now be written as

$$\begin{gathered}\mathtt{Pr}\left[\{i,j\} \text{ is generated from } P_D\right] = \\ \mathtt{Pr}\left[\phi\left(\mu\left(\chi\left(\varsigma_{f_D}(P_D)\right)\right), \mu\left(\chi\left(\varsigma_{f_D}(P_D)\right)\right)\right) = \{i,j\}\right].\end{gathered} \tag{2}$$

2.2 Simple Genetic Algorithms

In the simple GA (SGA), a new population P' of fixed size r over search space Ω for the next generation is built according to population P with

$$\begin{gathered}\mathtt{Pr}\left[\tau(P) = P'\right] = \\ \frac{r!}{\prod_{i\in\Omega}(\sum_{j\in P'}[i=j])!} \prod_{i\in P'} \mathtt{Pr}\left[i \text{ is generated from } P\right]\end{gathered} \tag{3}$$

where $\tau : \pi \to \pi$ represents the stochastic construction of a new population from and into population space π of the SGA, and $P'(i)$ denotes the number of individuals i in P'. Since the system to create a new generation P' only depends on the previous state P, the SGA is said to be Markovian. The SGA can now be written as a Markov chain with transition matrix T with $T_{P'P} = \mathtt{Pr}\left[\tau(P) = P'\right]$. If mutation can map any individual to any other individual, all elements of T become strictly positive, and T becomes irreducible and aperiodic. The limit behavior of the Markov chain can then be studied by finding the eigenvector, with corresponding eigenvalue 1, of T.

We will assume uniform crossover, bitwise mutation according to a mutation probability μ, and selection proportional to fitness throughout the paper.

This completes the formal construction of haploid and diploid simple genetic algorithms. More details of this construction can be found in [13].

2.3 Co-evolution of Finite Population Models

Next, we consider the combined co-evolutionary process of two SGAs, respectively defined by population transitions τ_1 and τ_2, over population search spaces π_1 and π_2. We assume that the population sizes of both algorithms are fixed and finite, and their generational transitions are executed at the same rate.

In order to make the representative GAs – and thus their fitness functions – interdependent, we need to override the fitness evaluation $f : \Omega \to \mathbb{R}^+$ of any one of the co-evolving GAs with $f_i : \Omega_i \times \pi_j \to \mathbb{R}^+$ where Ω_i is the search space of the GA, and π_j is the population state space of the co-evolving GA. As such, the fitness function of an individual in one population becomes dependent on the configuration of the population of the co-evolving GA. Consequently, the

generation probabilities of equation (3) now also depend on the population of the competing algorithm.

The state space π_{co} of the resulting Markov chain of the co-evolutionary algorithm is defined as the Cartesian product of spaces π_1 and π_2, i.e., $\pi_{co} = \pi_1 \times \pi_2$. All (P, Q), with $P \in \pi_1, Q \in \pi_2$, are states of the co-evolutionary algorithm. Generally, the transition $\tau_{co} : \pi_{co} \to \pi_{co}$ in the co-evolutionary Markov chain of two interdependent Markov chains is defined by

$$\mathtt{Pr}\left[\tau_{co}((P,Q)) = (P',Q')\right] = \mathtt{Pr}\left[\tau_1(P) = P'|Q\right] \cdot \mathtt{Pr}\left[\tau_2(Q) = Q'|P\right] \qquad (4)$$

where populations P and Q are states of π_1 and π_2 respectively. The dependence of τ_1 and τ_2 on Q and P respectively, gives way for the implementation of a coupled fitness function for either algorithm.

2.4 Limit Behavior

One can show that the combination of irreducible and aperiodic interdependent Markov chains, as defined above, does not generally result in an irreducible and aperiodic Markov chain. Therefore, we cannot simply assume that the Markov chain that defines the co-evolutionary process converges to a unique fixed point.

We can, however, make the following assumptions: If mutation can map any individual – in both of the co-evolving GAs – to any other individual in the algorithm's search space with a strictly positive probability, then all elements in the transition matrices of both co-evolving Markov chains are always nonzero and strictly positive. As a result from multiplying the transition probabilities in equation (4), all transition probabilities of the co-evolutionary Markov chain are thus strictly positive. This makes the combined Markov chain irreducible and aperiodic, such that the limit behavior of the whole co-evolutionary process can be studied by finding the unique eigenvector, with corresponding eigenvalue 1, of the transition matrix as defined by equation (4), due to the Perron-Frobenius theorem [14].

2.5 Expected Performance

The eigenvector, with corresponding eigenvalue 1, of the co-evolutionary Markov chain describes the fixed point distribution over all possible states (P, Q) of the Markov chain in the limit. As a result, toward the limit, the Markov chain converges to the distribution that describes the overall mean behavior of the co-evolutionary system. If a simulation is run that starts with an initial population according to this distribution, the distribution over the states at all next generations are also according to this fixed point distribution. For each of the states, we can compute the mean fitness of the constituent populations of that state. With this information, and the distribution over all states in the limit, we can make a weighted mean to find the mean fitness of both algorithms in the co-evolutionary system at hand.

More formally, let T denote the $|\pi_{co}| \times |\pi_{co}|$ transition matrix of the co-evolutionary system with transition probabilities

$$T_{(P',Q'),(P,Q)} = \mathtt{Pr}\left[\tau_{co}((P,Q)) = (P',Q')\right]$$

as defined by equation (4). Let ξ denote the eigenvector, with corresponding eigenvalue 1, of T. ξ denotes the distribution of states of the co-evolutionary algorithm in the limit, with component $\xi_{(P,Q)}$ denoting the probability of ending up in state $(P,Q) \in \pi_{co}$ in the limit. If $\overline{f_1}(P,Q)$ gives the mean fitness of the individuals in population P, given an opponent population Q, then

$$\overline{f_1} = \sum_{(P,Q)\in\pi_{co}} \xi_{(P,Q)} \cdot \overline{f_1}(P,Q) \quad \text{with} \quad \overline{f_1}(P,Q) = \frac{1}{|P|}\sum_{i\in P} f_1(i,Q), \tag{5}$$

gives the mean fitness of the populations governing the dynamics of the first algorithm toward the limit, in relation to its co-evolving algorithm. Similarly, the mean fitness of the second algorithm can be computed. We use the mean fitness in the limit as an exact measure of performance of the algorithm, in relation to the co-evolving algorithm. Equation (5) also gives the expected mean fitness of the co-evolving algorithms if simulations of the model are executed.

We will also calculate the variance and standard deviation in order to discuss the significance of the exact results. The variance of the fitness of the first algorithm, according to distribution ξ, is equal to

$$\sigma^2_{f_1} = \sum_{(P,Q)\in\pi_{co}} \xi_{(P,Q)} \cdot \left(\overline{f_1}(P,Q) - \overline{f_1}\right)^2. \tag{6}$$

Similarly to the mean fitness, the variance of the fitness gives an expectation of the variance for simulations of the model.

Given the parameters for fitness determination, selection and reproduction of both co-evolving GAs in the co-evolutionary system, we can now estimate the mean fitness, and discuss the performance of both genetic algorithms, in the context of their competitors' performance.

3 Application

3.1 Competitive Game: Matching Pennies

In order to construct interdependent fitness functions, we can borrow ideas of competitive games from Evolutionary Game Theory (EGT, overviews can be found in [15,16]). EGT studies the dynamics and equilibriums of games played by populations of players. The strategies players employ in the games determine their interdependent fitness.

A common model to study the dynamics – of frequencies of strategies adopted by the populations – is based upon replicator dynamics. This model makes a couple of assumptions, some of which will be discarded in our model. Replicator

dynamics assumes infinite populations, asexual reproduction, complete mixing, i.e., all players are equally likely to interact in the game, and strategies breed true, i.e., strategies are transmitted to offspring proportionally to the payoff achieved. In our finite population model, where two GAs compete against each other, we maintain the assumption that strategies breed true. We also maintain complete mixing, although the stochastic model also represents incomplete mixing with randomly chosen opponent strategies. We now consider finite fixed population sizes with variation and sexual reproduction of strategies.

In the scope of our application, we focus on a family of 2×2 games called "matching pennies." Consider the payoff matrices for the game in Table 1. Each of the two players in the game either calls 'heads' or 'tails.' Depending on the players' calls and their representative values in the payoff matrices, the players receive a payoff. More specifically, the first player receives payoff $1-L$ if the calls match, and L otherwise. The second player receives 1 minus the first player's payoff. If L ranges between 0 and 0.5, the first player's goal therefore is to call the same as the second player, whose goal in turn is to do the inverse. Hence the notion of competition in the game.

Table 1. Payoff matrices of the matching pennies game. One population uses payoff matrix f_1, where the other players use payoff matrix f_2. Parameter L denotes the payoff received when the player loses the game, and can range from 0 to 0.5

f_1	heads	tails
heads	$1-L$	L
tails	L	$1-L$

f_2	heads	tails
heads	L	$1-L$
tails	$1-L$	L

Let a population of players denote a finite sized population consisting of individuals who either call 'heads' or 'tails.' In our co-evolutionary setup, two GAs evolving such populations P and Q are put against one another. The fitnesses of individuals in population P and Q are based on f_1 and f_2, from Table 1, respectively. We use complete mixing to determine the fitness of each individual in either of the populations: Let p_{heads} denote the proportion of individuals in population P who call 'heads,' and q_{heads} the proportion of individuals in Q to call 'heads.' Define p_{tails} and q_{tails} similarly for the proportion of 'tails' in the populations. The fitness of an individual i of population P, regarding the constituent strategies of population Q, can now be defined as

$$f_1(i,Q) = \begin{cases} q_{\text{heads}} \cdot (1-L) + q_{\text{tails}} \cdot L & \text{if } i \text{ calls 'heads'} \\ q_{\text{tails}} \cdot (1-L) + q_{\text{heads}} \cdot L & \text{if } i \text{ calls 'tails'} \end{cases} \tag{7}$$

and that of an individual j in population Q as

$$f_2(j,P) = \begin{cases} p_{\text{heads}} \cdot L + p_{\text{tails}} \cdot (1-L) & \text{if } j \text{ calls 'heads'} \\ p_{\text{tails}} \cdot L + p_{\text{heads}} \cdot (1-L) & \text{if } j \text{ calls 'tails'} \end{cases} \tag{8}$$

It can easily be verified that the mean fitness of population P always equals 1 minus the mean fitness of population Q, i.e., $\overline{f_1}(P,Q) = 1 - \overline{f_2}(Q,P)$. Similarly, the mean fitness of both algorithms sum up to 1, with $\overline{f_1} = 1 - \overline{f_2}$, c.f. equation (5). If we assume $0 \leq L < 0.5$, then there exists a unique Nash equilibrium of this game, where both populations call 'heads' or 'tails,' each with probability 0.5. In this equilibrium, both populations receive a mean fitness of 0.5. No player can benefit by changing her strategy while the other players keep their strategies unchanged. Any deviation from this indicates that one algorithm relatively performs better at the co-evolutionary task at hand than the other. As we want to compare the performance of algorithms in a competitive co-evolutionary setup, this is a viable null hypothesis.

3.2 Haploid versus Diploid

For the matching pennies game, we construct a co-evolutionary Markov chain in which a haploid and diploid GA compete with each other. With this construction, and their transition matrices, we can determine the performance of both algorithms according to the limit behavior of the Markov chain. Depending on the results, either algorithm can be elected as a relatively better algorithm.

Let the length of binary strings in both algorithms be $l = 1$. This is referred to as the single locus, two allele problem, a common, yet small, setup in population genetics. An individual with phenotype 0 calls 'heads,' and 'tails' if the phenotype is 1. Note that uniform crossover will not recombine genes since there is only one locus, but will rather select one of both parent gametes.

Let π_{co} be the search space of the co-evolutionary system, defined by the Cartesian product of the haploid populations' search space π_H and diploid populations π_D, such that $\pi_{co} = \pi_H \times \pi_D$. Depending on a fixed population size r for both competing algorithms, $|\pi_{co}| = ((r+2)(r+1)^2)/2$ denotes the size of the co-evolutionary state space.

For any state $(P,Q) \in \pi_{co}$, let equations (7) and (8) be the respective fitness functions for the individuals in the haploid and diploid algorithms. Since we want to compare the algorithms' performance under comparable conditions, both populations are assumed to have the same parameters for recombination and mutation.

3.3 Limit Behavior and Mean Fitness

According to the definition of the co-evolutionary system in equation (4), the transition matrix for a given set of parameters can be calculated. The eigenvector, with corresponding eigenvalue 1, of this transition matrix can be found through iterated multiplication of the transition matrix with an initially distributed stochastic vector. From the resulting eigenvector we can find the mean fitness of the co-evolutionary GAs toward the limit. These means are discussed in the following sections.

We split the presentation of the limit behavior results into two separate sections. In the first section, we discuss the results given the assumption of pure

dominance, i.e., one of both alleles, either 0 or 1 is strictly dominant over the other allele. In the second part, we discuss the results in the case of partial dominance. In this setting, the phenotype of the diploid heterozygous genotype $\{0, 1\}$ is defined by a probability distribution over 0 and 1.

Pure dominance. Let 1 be the dominant allele, and 0 the recessive allele in diploid heterozygous individuals. This implies that diploid individuals with genotype $\{0, 1\}$ have phenotype 1 [1].

Figure 1 shows the mean fitness of the haploid algorithm, which is derived from the co-evolutionary systems' limit behavior, using equation (5). The proportion of parameter settings for which diploidy performs better, increases as the population size of the algorithms becomes bigger.

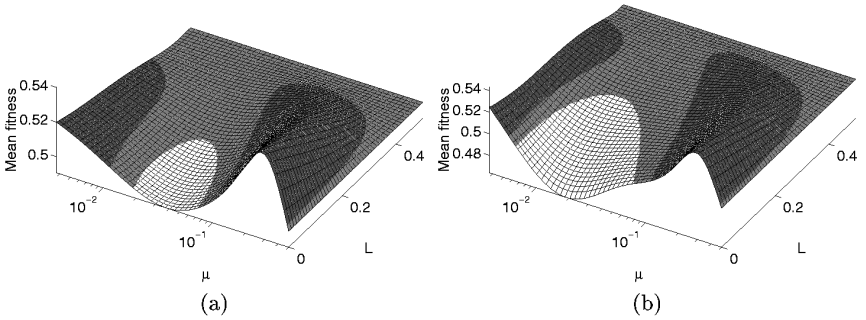

Fig. 1. Exact mean fitness of the haploid GA in the co-evolutionary system, for variable mutation rate μ and payoff parameter L. The mean fitness of the diploid algorithm always equals 1 minus the mean fitness of the haploid algorithm. Population size of both algorithms is fixed to 5 in (a) and 15 in (b). The mesh is colored light as the mean fitness is below 0.4975, i.e. when the diploid algorithm performs better, and dark as the mean fitness is over 0.5025, i.e. for parameters where haploidy performs better.

In our computations, we found a fairly large standard deviation near $\mu = 0$ and $L = 0$. The standard deviation goes to zero as either of the parameters go to 0.5. We discuss the source of this fairly large standard deviation in section 3.4. Because of the large standard deviation, it is very hard to obtain these results with empirical runs of the model. However, it is hard to compute the exact limit behavior of large population systems, since this implies that we need to find the eigenvector of a matrix with $\mathcal{O}(r^6)$ elements for population size r.

Partial dominance. Instead of using a pure dominance scheme in the diploid GA, we can also assign a partial dominance scheme to the dominance operator. In

[1] If we would choose 0 as the dominant allele instead of 1, the co-evolutionary system would yield the exact same performance results, because of symmetries in the matching pennies game. The same holds for exchanging fitness functions f_1 and f_2.

this dominance scheme, the heterozygous genotype $\{0, 1\}$ has phenotype 0 with probability h, and phenotype 1 with probability $1-h$. h is called the dominance degree or coefficient. The dominance degree is the measure of dominance of the recessive allele in the case of heterozygosity. Since our model is stochastic, we could also state that the fitness of an heterozygous individual is an intermediate of the fitnesses of both homozygous phenotypes.

The performance results are summarized in Figure 2. The figures show significantly better performance results for the diploid algorithm under small mutation and high selection pressure (small L), in relation to the haploid algorithm. Indeed, if we consider partial dominance instead of pure dominance, the memorized strategies in the recessive alleles of a partial dominant diploid population are tested against the environment, even in heterozygous individuals. The fact that this could lead to lower fitnesses in heterozygous individuals because of interpolation of high and low fitness, does not restrict the diploid algorithm from obtaining a higher mean fitness in the co-evolutionary algorithm. The standard deviation is smaller than in the pure dominance case. This is explained in section 3.4.

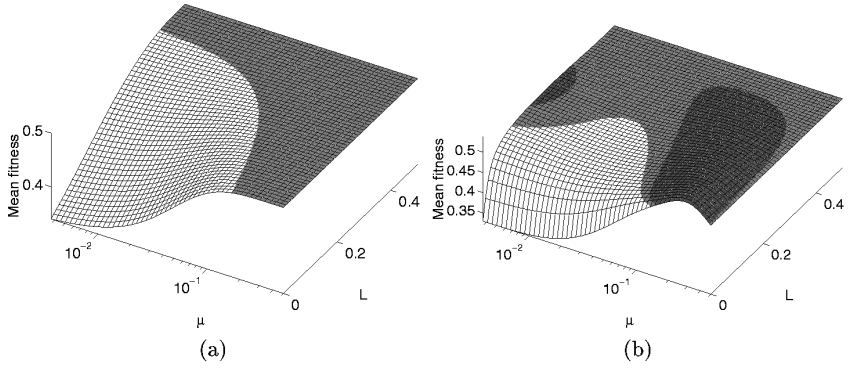

Fig. 2. Mean fitness in the limit of the haploid algorithm similar to Figure 1 for different dominance coefficients, with $r = 15$. Figure (a) applies dominance degree $h = 0.5$ and (b) has dominance degree $h = 0.01$. Figure 1 applies dominance degree $h = 0$

3.4 Source of High Variance

In order to find where the high variance originates, we analyze the distribution of fitness at the fixed point. Dissecting the stable fixed point shows that there are a small number of states with high probability, and many other states with a small probability. More specifically, of these states with a high probability, about half of them have an extremely high mean fitness for one algorithm, where the other half have an extremely low mean fitness. This explains the high variance in the fitness distribution. If we would run a simulation of the model, we would

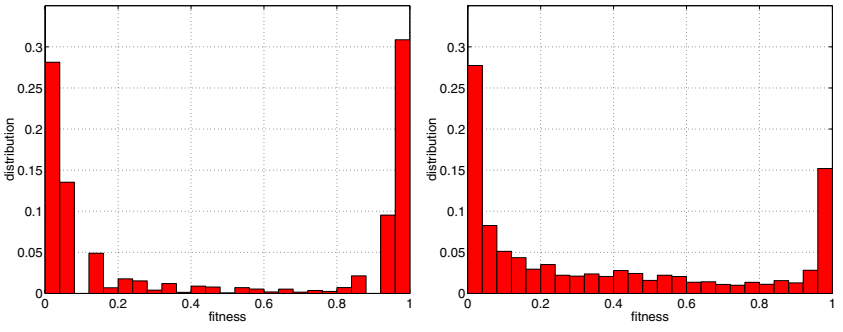

Fig. 3. Histogram showing the distribution of fitness of the haploid genetic algorithm, in the limit. Both figures have parameters $r = 10, \mu = 0.01, L = 0$. Figure (a) shows the distribution for $h = 0$ and $h = 0.5$ for (b). $\overline{f_1} = 0.4768$ and $\sigma_{f_1} = 0.4528$ in histogram (a) and $\overline{f_1} = 0.3699$ and $\sigma_{f_1} = 0.3715$ in (b)

see that the algorithm alternately visits high and low fitness states, and switches relatively fast between these sets of states.

Figure 3 shows that, toward the limit, the mean fitness largely depends on states with both extremely low and high fitnesses, which corresponds with the high standard deviation. Note that the standard deviation is smaller in the case of a higher dominance degree. This is also due to average fitnesses being smeared out in heterozygous individuals because of the higher dominance degree. The relative difference between frequencies of extremely low and high fitnesses also results in a lower variance, as the dominance degree increases.

4 Discussion

This paper shows how a co-evolutionary model of two GAs with finite population size can be constructed. We also provide ways to measure and discuss the relative performance of the algorithms at hand. Because of the use of Markov chains, exact stochastic results can be computed.

The analyses presented in the application of this paper show that, given the matching pennies game, and if pure dominance is assumed, the results are only in favor of diploidy in case of specific parameter settings. Even then, the results are not significant and subject to a large standard deviation. A diploid GA with partial dominance and a strictly positive dominance degree can outperform a haploid GA, if similar conditions hold for both algorithms. These results are expressed best under low mutation pressure and high selection pressure, i.e., when a deleterious mutation has an almost lethal effect on the individual. Diploidy performs relatively better as the population size increases.

Based on these results, we suggest that further research should be undertaken on the usage of diploidy in co-evolutionary GAs. This paper studies a

small problem and small search spaces. Empirical evidence might prove to be a useful tool in studying complexer problems, or larger populations. Scaled up versions – of small situations which can be analyzed exactly – could be used as empirical evidence to support exact predictions. Low significance and high standard deviations might prove that the study of relative performance of GAs in competitive co-evolutionary situations is, however, empirically hard.

References

1. S. G. Ficici, O. Melnik, and J. B. Pollack. A game-theoretic investigation of selection methods used in evolutionary algorithms. In *Proceedings of the 2000 Congress on Evolutionary Computation*, 2000.
2. S. G. Ficici and J. B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In *Parallel Problem Solving from Nature VI*, 2000.
3. C. D. Rosin. *Coevolutionary search among adversaries.* PhD thesis, San Diego, CA, 1997.
4. A. Lubberts and R. Miikkulainen. Co-evolving a go-playing neural network. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 14–19, 2001.
5. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In *Artificial Life II.* Addison Wesley, 1992.
6. D. Floreano, F. Mondada, and S. Nolfi. Co-evolution and ontogenetic change in competing robots. In *Robotics and Autonomous Systems*, 1999.
7. D. E Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *Second International Conference on Genetic Algorithms*, pages 59–68, 1987.
8. J. Lewis, E. Hart, and G. Ritchie. A comparison of dominance mechanisms and simple mutation on non-stationary problems. In *Parallel Problem Solving from Nature V*, pages 139–148, 1998.
9. K. P. Ng and K. C. Wong. A new diploid scheme and dominance change mechanism for non-stationary function optimization. In *6th Int. Conf. on Genetic Algorithms*, pages 159–166, 1995.
10. R. E. Smith and D. E. Goldberg. Diploidy and dominance in artificial genetic search. *Complex Systems*, 6:251–285, 1992.
11. A. M. L. Liekens, H. M. M. ten Eikelder, and P. A. J. Hilbers. Finite population models of dynamic optimization with alternating fitness functions. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, 2003.
12. A. E. Nix and M. D. Vose. Modelling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, pages 79–88, 1992.
13. A. M. L. Liekens, H. M. M. ten Eikelder, and P. A. J. Hilbers. Modeling and simulating diploid simple genetic algorithms. In *Foundations of Genetic Algorithms VII*, 2003.
14. D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation.* Prentice-Hall, 1989.
15. J. W. Weibull. *Evolutionary Game Theory.* MIT Press, Cambridge, Massachusetts, 1995.
16. J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics.* Cambridge University Press, 1998.