Measuring Generalization Performance in Co-evolutionary Learning

Siang Y. Chong, Peter Tiňo, and Xin Yao
The Centre of Excellence for Research in
Computational Intelligence and Applications (CERCIA)
School of Computer Science
The University of Birmingham, UK.
E-mails: S.Y.Chong@cs.bham.ac.uk, P.Tino@cs.bham.ac.uk, X.Yao@cs.bham.ac.uk

Abstract

Co-evolutionary learning involves a training process where training samples are instances of solutions that interact strategically to guide the evolutionary (learning) process. One main research issue is with the generalization performance, i.e., the search for solutions (e.g., input-output mappings) that best predict the required output for any new input that has not been seen during the evolutionary process. However, there is currently no such framework for determining the generalization performance in co-evolutionary learning even though the notion of generalization is well-understood in machine learning. In this paper, we introduce a theoretical framework to address this research issue. We present the framework in terms of game-playing although our results are more general. Here, a strategy's generalization performance is its average performance against all test strategies. Given that the true value may not be determined by solving analytically a closed-form formula and is computationally prohibitive, we propose an estimation procedure that computes the average performance against a small sample of random test strategies instead. We perform a mathematical analysis to provide a statistical claim on the accuracy of our estimation procedure, which can be further improved by performing a second estimation on the variance of the random variable. For game-playing, it is well-known that one is more interested in the generalization performance against a biased and diverse sample of "good" test strategies. We introduce a simple approach to obtain such a test sample through the multiple partial enumerative search of the strategy space that does not require human expertise and is generally applicable to a wide range of domains. We illustrate the generalization framework on the co-evolutionary learning of the iterated prisoner's dilemma (IPD) games. We investigate two definitions of generalization performance for the IPD game based on different performance criteria, e.g., in terms of the number of wins based on individual outcomes and in terms of average payoff. We show that a small sample of test strategies can be used to estimate the generalization performance. We also show that the generalization performance using a biased and diverse set of "good" test strategies is lower compared to the unbiased case for the IPD game. This is the first time that generalization is defined and analyzed rigorously in co-evolutionary learning. The framework allows the evaluation of the generalization performance of any co-evolutionary learning system quantitatively.

Keywords: Evolutionary computation, co-evolutionary learning, generalization, Chebyshev's inequality, iterated prisoner's dilemma.

1 Introduction

Co-evolutionary learning refers to a broad class of population-based, stochastic search algorithms that involve the simultaneous evolution of competing solutions with coupled fitness [1]. A co-evolutionary learning system can be implemented using co-evolutionary algorithms [2, 3], which can be derived from evolutionary algorithms (EAs) [4, 5]. That is, both co-evolutionary learning and EAs can be described in terms of the framework whereby an adaptation process is carried out on the solutions in some form of representation through a repeated process of variation and selection. The framework distinguishes co-evolutionary learning and EAs in general from classical approaches (e.g., steepest-descent-based algorithms) from two specific features, i.e., they are population-based and incorporate information exchange

mechanisms between populations in successive generations (iterative steps) to guide the search process [6].

Despite their similarity in framework, co-evolutionary learning and EAs are fundamentally different in how the fitness of a solution is assigned, leading to significantly different outcomes when applied to similar problems (e.g., different search behaviors on the space of solutions [7, 8]). EAs are often viewed and constructed in terms of an *optimization* context [4, 5], whereby an absolute fitness function is required to assign the fitness value to a solution that is *objective* (fitness value for a solution is always the same regardless of the context). For co-evolutionary learning, the fitness of a solution is obtained through its interactions with other competing solutions in the current population, and as such, is *subjective* (fitness value for a solution depends on the context, e.g., population, and as such is *relative* and *dynamic*). Here, co-evolutionary learning operates to find solutions guided by strategic interactions among the competing solutions from one generation to the next that results (hopefully) in an *arms race* of increasingly innovative solutions [9, 10, 11].

One of the early motivations for using co-evolutionary learning is its potential application for solving problems that cannot be framed in the context of optimization (e.g., using EAs) because it is not possible or very difficult to formulate an absolute fitness function that reflects the underlying properties of the problem. For such problems, continued use of an inappropriate fitness function will often bias the search to solutions that do not reflect the underlying properties of the problem, leading to suboptimal solutions [12]. Even if a fitness function can be formulated, it may not be able to evaluate and differentiate between individual solutions to provide some gradient to direct the search when using EAs [8, 13]. One such problem that is difficult to solve using EAs, but can be naturally framed in co-evolutionary learning, is the problem of game-playing [2, 3, 14, 15, 16, 17].

However, despite the success of co-evolutionary learning in solving the problem of games [2, 3, 14, 15, 16, 17] (and other problems in the context of optimization [18] and classification [19, 20]), the approach is not trouble-free for all problems. In particular, co-evolutionary learning is now recognized to suffer from problems (collectively called *co-evolutionary pathologies*) that affect the performance of a co-evolutionary learning system [11, 12, 13, 21, 22, 23]. For example, *overspecialization* to a single solution can occur in the population [21], which can be a result of earlier population *disengagement*, e.g., some solutions are favored over others due to large gaps in competence level [13]. When intransitivity exists in the relationship between solutions, *cyclic dynamics* may occur during co-evolution, whereby at some point in the process, the population overspecializes to a solution that is vulnerable to another solution that exploits it [11, 12]. Furthermore, when a solution is driven to extinction but at a later point is adaptively found again, the co-evolution is said to exhibit *forgetting* [22, 23].

Co-evolutionary pathologies are usually attributed to the use of relative fitness in the selection process of a co-evolutionary learning system [8, 24]. There are also broader implications in the design of the co-evolutionary learning system to its performance that is dependent on the co-evolutionary search dynamics (i.e., representation [25], variation [21, 25, 26], and selection [27]). This necessitates a more in-depth study of co-evolutionary learning search dynamics. In particular, one approach that has been given much attention in the past is the monitoring of the progress of the co-evolution for search of more innovative and sophisticated solutions, e.g., arms race dynamics [28, 29, 30, 31].

Another approach is to consider a global view of increasing performance in co-evolutionary learning. However, for this particular investigation, previous studies are restricted to simple problems or problems where the global view is known in advance [8, 11]. Tools introduced for monitoring progress of arms race dynamics are inappropriate or may not be suitably adapted for the global view analysis because they only provide relative performance information of solutions between different generations. This is because solutions in the current generation that are better than those in previous generations do not necessarily imply that they perform better globally when compared with new or all possible solutions.

In machine learning, there exists a powerful framework, generalization, that provides a global view of performance for learning systems. Generalization refers to the ability of the learning system to find the solution, which can be viewed in the context of input-output mappings, that best predicts the required output for any new input that has not been seen during the training process. With the context of generalization, one is interested in how the learning system can realize the underlying properties of the problem from a small sample of training data (e.g., the input-output set) to produce the solution. For example, in the case of neural network training, one is not interested in learning the "exact representation

of the training data itself, but rather to build a statistical model of the process which generates the data" (page 332 of [32]). Furthermore, it should be noted that learning is not necessarily the same as optimizing because the solution with optimal fitness does not necessarily imply it has the best generalization (unless an accurate metric to measure generalization is used in the fitness function) [33].

Co-evolutionary learning, like other learning systems, also uses a small training sample during the evolutionary (training) process to produce a solution to the problem. However, co-evolutionary learning is different from other learning systems in that the training samples are not fixed, but instead, are instances of solutions that are changing (evolving) and are interacting with each other strategically to guide the evolutionary process (i.e., learning). Despite this difference, the generalization framework can be applied to co-evolutionary learning systems to provide a global view in performance. Although the notion of generalization is well-understood in machine learning, there is a lack of theoretical framework to justify how the generalization performance of a co-evolutionary learning system is determined with respect to the problem. For example, past studies such as [33, 34] have used a large sample of randomly obtained test cases to estimate the generalization performance. It is not known how accurate such an estimation is, i.e., how close the estimated generalization performance (using test samples randomly obtained from the search space) to the true generalization performance (using the entire search space).

In this paper, we introduce a theoretical framework that addresses the problem of determining the generalization performance of co-evolutionary learning in general. We present the framework in terms of game-playing, i.e., learning game strategies that generalize well to the game (e.g., defeat a large number of strategies that exist). However, our theoretical framework is general, i.e., the problem can be put in the context of test-based evaluations (e.g., comparisons between solutions) where some tests can reflect the underlying properties (objectives) of the problem that are unknown (game-playing is one such problem) [12]. We first define *generalization performance* of a strategy as its *average performance against all test strategies*. With this definition, it follows that the best generalization performance for a co-evolutionary learning system is the one that produces evolved strategies with the maximum average performance against all strategies.

Although this definition is simple, the generalization performance can be difficult to determine due to two reasons. First, the analytical function for game outcomes can be unknown, and as such, we cannot determine the generalization performance by solving analytically a closed-form formula. Second, the strategy space can be very large (although finite), thus making it computationally prohibitive. To address this problem, we propose the alternative of *estimating* the generalization performance by taking the average performance of the evolved strategy against a sample of test strategies that are randomly drawn from the strategy space.

We show through a mathematical analysis using Chebyshev's Theorem that the probability that the absolute difference between the estimated and true values exceeding a given error (precision value) is bounded by a value that reciprocally depends only on the square of the error and the size of the random test sample. However, this probability bound assumes the worst-case of having maximum variance for the distribution of the random variable over a bounded interval. In general, the true variance is smaller than the maximum value. As such, we perform a mathematical analysis and show how a second estimation of the variance can be used to obtain a tighter bound. In addition, we also show that for some games, the true variance of a strategy performance with respect to the strategy space is smaller, and as such, requires a smaller sample size to make the same statistical claim.

With this framework, it is now shown that a sample of randomly obtained test strategies that is much smaller in size compared with the total number of strategies in the space is sufficient to estimate the generalization performance of co-evolutionary learning, and that this estimation is close enough to the true value. We apply the framework to the co-evolutionary learning of the IPD games to illustrate both the advantage of the framework and how it can be used in general. In particular, we investigate two different definitions of generalization performance for the IPD game based on different performance criteria, e.g., in terms of the number of wins based on individual outcomes and in terms of average payoff.

It is well-known that rather than considering the average performance for all cases, one may be more interested in the average performance for specific cases that are more common or that arise naturally. In the context of game-playing, one is more interested in the generalization performance of the co-evolutionary learning system against "good" strategies, and not the average performance against all strategies since it is possible that a large proportion of strategies in the space are poor or mediocre. To

determine generalization performance against this biased sample of good test strategies, we introduce a simple approach to obtain such a test sample through the multiple partial enumerative search of the strategy space. Each partial enumerative search uses a population size that is much larger than the total number of possible unique strategies that can be searched during co-evolution to produce a best strategy. This approach does not require human expertise (as in generating some arbitrarily designed strategies) and is more comprehensive than the previous single partial enumerative search that we introduced earlier in [35, 36, 37]. We show that for the co-evolutionary learning of IPD games, the generalization performance for the case of a biased and diverse sample is lower compared to the case of an unbiased sample.

This paper is organized as follows. Section 2 presents our theoretical framework of generalization performance of co-evolutionary learning. Section 3 illustrates how the framework can be applied to the IPD game. Section 4 investigates the application of the framework to estimate the generalization performance of co-evolutionary learning of simple IPD games where the true generalization performance can be determined. Section 5 presents an empirical study on estimating generalization performance of co-evolutionary learning of slightly more complex games where the true generalization performance cannot be determined. The section also compares the results of estimates based on using an unbiased test sample and that of using a biased and diverse sample of "good" test strategies. Section 6 discusses the implications of the framework and concludes the paper.

2 Generalization Framework for Co-evolutionary Leaning

2.1 A Need for a Consistent and General Approach to Estimate the Generalization Performance of Co-evolutionary Learning

Darwen and Yao [33, 34, 35] were among the first to explicitly investigate co-evolutionary learning through the framework of generalization from machine learning (others include [38]). However, they [33, 34, 35] studied the issue through an empirical approach only. In particular, they investigated the utility of using some random sample of test cases to estimate the generalization performance of a co-evolutionary learning system.

There are other studies that can be related to generalization in the context of co-evolutionary learning. Wiegand and Potter [39] studied the notion of robustness (of individual components) in a cooperative co-evolution setting. Ficici and Pollack [23] studied the notion of *solution concepts*, e.g., a *partitioning* of search space into solutions (that are wanted) and non-solutions for a problem from measuring some properties and establishing some criteria that the searched point is a solution. Bowling [40] studied the notion of *regret*, i.e., measures performance difference between a learning algorithm with the best static strategy during training. Powers and Shoham [41] studied the estimation of best-response through the use of random samples. Studies in [42, 43, 44] investigated formalisms of monotonic improvement in co-evolutionary learning and developed algorithmic frameworks that guarantee monotonic improvements of co-evolving solutions based on archive of test cases.

Here, our main motivation is to develop a framework for a rigorous quantitative analysis of performance in co-evolutionary learning using the notion of generalization from machine learning. We are motivated to address the need for a principled approach to estimate the generalization performance of co-evolutionary learning. The framework aims to allow one to estimate the generalization performance in general for problems co-evolutionary learning is used to solve and at any point in the evolutionary process where generalization performance is measured.

There are two reasons why measuring generalization performance of co-evolutionary learning is necessary and important. First, it is used to provide an absolute quality measure on how well a co-evolutionary learning system is performing with respect to the problem, i.e., how well the co-evolutionary learning generalizes. Second, it can be used as a means to compare the generalization performance of different co-evolutionary learning systems with respect to the problem.

We first introduce a theoretical framework that defines explicitly the generalization performance for co-evolutionary learning, and how it can be determined, i.e., measured. However, it is noted that obtaining

the true generalization performance for a co-evolutionary learning system is very difficult. As such, through the theoretical framework, we provide an alternative of a consistent and general procedure to estimate the generalization performance. We show a mathematical analysis of how the generalization performance can be estimated using a random sample of test cases. We demonstrate the utility of the estimation procedure by determining the statistical claim that one can make of how confident one is with the accuracy of the estimate compared to the true generalization for a random test sample of a given size.

2.2 Estimating Generalization Performance

In co-evolutionary learning, the quality of a solution is determined relative to other solutions. This is achieved through comparisons, i.e., interactions between solutions. These interactions can be framed in terms of game-playing, i.e., an interaction is a game played between two strategies (solutions). The game outcome of a strategy i against the opponent strategy j is $G_i(j)$, and conversely, the game outcome of strategy j against strategy i is denoted $G_j(i)$. Strategy i is said to solve the test provided by strategy j if $G_i(j) \geq G_j(i)$[1].

Here, the absolute quality (generalization performance) of a strategy i is measured with respect to its expected performance in solving tests provided by strategies j. The goal of co-evolutionary learning for the problem of game-playing is to learn a strategy i with the best generalization performance. In the following, we present a theoretical framework for estimating the generalization performance of co-evolutionary learning. It should be noted that the framework is presented in the context of the complete solution. As such, the framework is directly applicable for the estimation of generalization performance of a complete solution obtained either by a competitive or a cooperative co-evolutionary learning system.

2.2.1 True Generalization Performance

The generalization performance of co-evolutionary learning is determined with respect to the evolved strategy that is produced. The true generalization performance of co-evolutionary learning is defined as the expected performance of strategy i that is produced after a learning process (co-evolution) against all strategies j in the strategy space $\mathbf{S}$. The true generalization performance of strategy i, G_i, can be written as follows:

$$G_i = E_{P_1(j)}[G_i(j)] = \int_{\mathbf{S}} G_i(j) P_1(j) \mathrm{d}j, \tag{1}$$

where G_i is the expectation of strategy's i performance against j, $G_i(j)$, with respect to the distribution $P_1(j)$ over strategy space $\mathbf{S}$ (i.e., the distribution with which opponent strategies j are drawn). Note that this definition of generalization does not imply having to compare with all strategies that exist. Instead, $P_1(j)$, which can be specified by the strategy representation (e.g., neural networks), can be induced to the strategy space $\mathbf{S}$. As such, some strategies j that exist may not be included to determine G_i because $P_1(j) = 0$.

There are two difficulties to apply Equation 1 directly. First, the analytical form for $G_i(j)$ is not known or difficult to obtain (even for simple games). Second, the distribution $P_1(j)$ over $\mathbf{S}$ may be unknown (at best we can only sample from $\mathbf{S}$ from a strategy representation that is used).

However, it is possible for these games to have a finite number of possible unique strategies, e.g., $\mathbf{S}$ is discrete and finite, or that we can consider a subset of $\mathbf{S}$ that is discrete and finite by inducing some strategy distribution $P_1(j)$ to $\mathbf{S}$. For the purpose of presentation and simplicity, we assume a uniform strategy distribution in $\mathbf{S}$.

With this, one can compute the true generalization performance of co-evolutionary learning through a strategy i, which is simply its average performance against all strategies j, and can be written in the

[1]For example, in a zero-sum game such as chess, one can say that strategy i solves the test provided by strategy j if i *defeats* j, i.e., $G_i(j) > G_j(i)$.

form:

$$G_i = \frac{1}{M} \sum_{j}^{M} G_i(j), \tag{2}$$

where M is the total number of unique strategies j that strategy i plays against. For example, consider the *two*-choice IPD game with deterministic and reactive memory-one strategies, where each strategy consider its previous moves and opponent's previous move. For such a game, there is only a total of $2^{2^2+1} = 2^5 = 32$ unique strategies in the strategy space.

However, the true generalization performance may not be computed by applying Equation 2 as the game increases in complexity. For example, considering the IPD game above, if strategies can play from n choices (e.g., n-choice IPD game), the total number of unique strategies increases to n^{n^2+1}. As such, even a moderate increase in the number of choices result in a large increase in the total number of strategies. With three choices, there are $3^{3^2+1} = 3^{10} = 59049$ strategies. As the number of choices increase to four, the total number of strategies increases to $4^{4^2+1} = 4^{17} = 17179869184$, and so forth. Thus, computing the true generalization performance using Equation 2 quickly becomes computationally infeasible even for a small increase in the number of choices.

2.2.2 Estimated Generalization Performance

Given that the true generalization performance cannot be obtained (e.g., it cannot be determined by solving analytically a closed-form formula and is computationally prohibitive), the next alternative is to estimate the generalization performance. Here, we propose estimating the generalization performance by considering the average performance against a smaller sample of N test strategies that can be computed (i.e., $N \ll M$). The estimated generalization performance of strategy i is given by:

$$\hat{G}_i(S_N) = \frac{1}{N} \sum_{j \in S_N} G_i(j), \tag{3}$$

where S_N is the sample of N test strategies randomly drawn from $\mathbf{S}$. For the rest of this paper, we use the notation $\hat{G}_i$ to represent $\hat{G}_i(S_N)$.

2.2.3 Chebyshev's Bound for Determining the Accuracy of the Estimation of Generalization Performance

We want to know how close $\hat{G}_i$, obtained using a set of N test strategies randomly drawn from $\mathbf{S}$ of size N, when compared to G_i, which is obtained using all possible strategies from $\mathbf{S}$ of size M (where $M \gg N$). That is, we want to know if the absolute difference between $\hat{G}_i$ and G_i, i.e., $|\hat{G}_i - G_i|$, is sufficiently small, i.e., does not exceed a small positive number ϵ (where $\epsilon > 0$). If this is true, then we say $\hat{G}_i$ is approximately G_i, i.e., $\hat{G}_i \approx G_i$.

However, we do not know what $|\hat{G}_i - G_i|$ is, because G_i is not known. One can try to find how *likely* that $\hat{G}_i$ is *close* enough to G_i. That is, what is the probability that $|\hat{G}_i - G_i| \geq \epsilon$ if one uses N randomly obtained test strategies to obtain $\hat{G}_i$. We answer this question using *Chebyshev's Theorem* [45].

Our mathematical analysis does not assume a particular strategy distribution $P_1(j)$, which can be induced on $\mathbf{S}$ by a particular strategy representation. All that is required is that S_N is a set of independent identically distributed (i.i.d) strategies drawn from $\mathbf{S}$ according to $P_1(j)$, e.g., using the same strategy representation.

We want to estimate the difference, $D_N = \hat{G}_i - G_i$, which is a random variable because the set of strategies S_N used to obtain $\hat{G}_i$ is sampled randomly from $\mathbf{S}$. From the definition of D_N, we apply Chebyshev's Theorem [45] and obtain[2]:

[2]The analysis can be extended to games with probabilistic strategies as well since D_N is a random variable.

$$P(|\hat{G}_i - G_i| \geq \epsilon) \leq \frac{\sigma_i^2}{N \cdot \epsilon^2} \tag{4}$$

for a strategy i (for full details, please refer to the Appendix section). For the rest of this paper, we use the notation σ^2 to represent σ_i^2. Furthermore,

$$\begin{aligned}
\sigma^2 &= E_{P_1(j)}[g_j^2] \\
&= E_{P_1(j)}\Big[\Big(G_i(j) - G_i\Big)^2\Big] \\
&= E_{P_1(j)}\Big[\Big(G_i(j) - E_{P_1(j)}[G_i(j)]\Big)^2\Big] \\
&= \mathrm{Var}_{P_1(j)}[G_i(j)].
\end{aligned}$$

We can bound σ^2. In general, for the random variable $G_i(j)$ distributed over the interval $[G_{\mathrm{MIN}}, G_{\mathrm{MAX}}]$ (where G_{MAX} and G_{MIN} is the maximum and minimum of $G_i(j)$, respectively), the maximum variance σ^2_{MAX} is when half of the mass is at G_{MIN} and the other half is at G_{MAX}, which results in a mean of $(G_{\mathrm{MIN}} + G_{\mathrm{MAX}})/2$ and standard deviation of $\sigma_{\mathrm{MAX}} = (G_{\mathrm{MAX}} + G_{\mathrm{MIN}})/2$ (see [46]).

Let $R = G_{\mathrm{MAX}} - G_{\mathrm{MIN}}$ be the range of the random variable $G_i(j)$. We can thus restate Chebyshev's bound as Lemma 1 [45].

Lemma 1 *For a strategy i, let $\hat{G}_i$ be the estimated generalization performance with respect to N random test strategies and G_i be the true generalization performance. Consider the absolute difference $|\hat{G}_i - G_i|$, which is a random variable with distribution P_N taken on a compact interval $[G_{\mathrm{MIN}}, G_{\mathrm{MAX}}]$ of length $R = G_{\mathrm{MAX}} - G_{\mathrm{MIN}}$. Then, for any positive number $\epsilon > 0$:*

$$P_N(|\hat{G}_i - G_i| \geq \epsilon) \leq \frac{R^2}{4N \cdot \epsilon^2}. \tag{5}$$

We note the generality of this framework by observing the following three points:

1. The framework is *independent* of the complexity of the game, since it is independent of the size of the strategy space, and independent of the strategy distribution in the strategy space.
2. Both G_{MAX} and G_{MIN} are often known *a priori* as they are specified by the game, which means that we can always obtain an upper bound on (5).
3. The framework is *independent* of learning algorithms since the bound holds for any strategy in the strategy space.

2.2.4 Application of Chebyshev's Bound in the Generalization Framework

To see how Chebyshev's bound can be used in the framework for estimating generalization performance, we first need to identify the relationship given by Equation 5 between the probability $P(|\hat{G}_i - G_i| \geq \epsilon)$, the sample size N, and the precision value ϵ. For simplicity, let $\epsilon' = \epsilon/R$. This lets us simplify Equation 5 to:

$$P(|D_N|' \geq \epsilon') \leq \frac{1}{4N \cdot \epsilon'^2},$$

which is the same as

$$P(|D_N|' \leq \epsilon') \geq 1 - \frac{1}{4N \cdot \epsilon'^2},$$

where $|D_N|' = |D_N|/R = |\hat{G}_i - G_i|/R$ is the normalized absolute difference of estimated and true values of the generalization performance.

To illustrate the Chebyshev's bound further, we plot the curve of $P_N(\epsilon') = 1/(4N\epsilon'^2)$ as points in the P-ϵ' space. Figure 1 shows the curves P_N for Chebyshev's bounds for different Ns over a range of ϵ's in $[0.01, 0.1]$. Each curve P_N gives the upper bound for $P(|D_N|' \geq \epsilon')$ when a random sample of test strategies of size N is used to estimate the generalization performance.

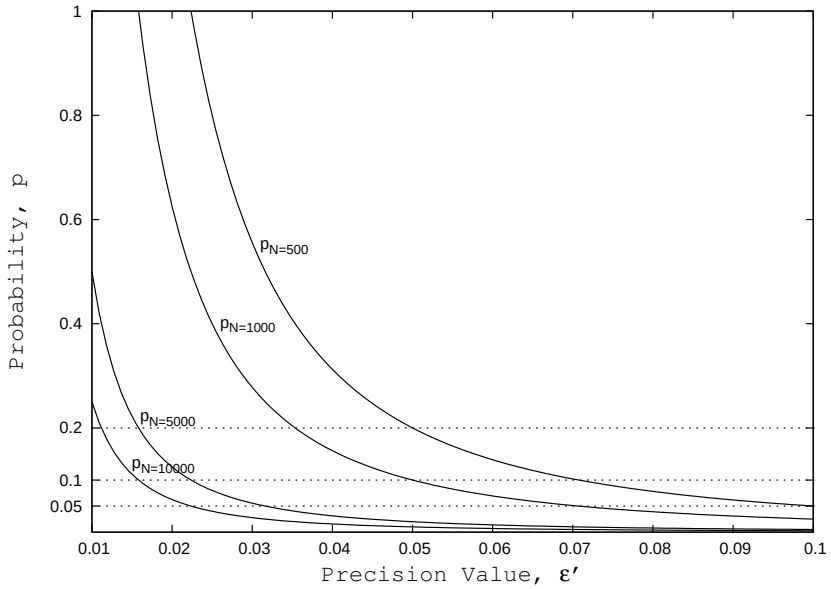


Figure 1: Figure showing various graphs of the theoretical Chebyshev's bounds in probability (given by Equation 5) P_N for different size N of the sample of random test strategies N in the range of precision value ϵ' in $[0.01, 0.1]$. Graphs are labelled as P_N.

Note that $P_N(\epsilon')$ is inversely proportional to N and ϵ'. For any ϵ', an increasingly larger number for N is required to obtain lower Chebyshev's bounds (Fig. 1). Assuming the worst-case, and when all the three values N, ϵ', and $P_N(\epsilon')$ are known, one can use a statistical argument to claim with confidence or probability at least $1 - P_N(\epsilon')$ that $|D_N|' \leq \epsilon'$ for a precision value ϵ'.

In practice, we only need to specify N to be as large as possible for which we can compute the estimated generalization performance. After that, we have to make a "tradeoff" between the confidence level (i.e., being more confident that the estimation is likely to be close to the true result) and precision value (i.e., the value by which the estimation will not be bigger or smaller compared to the true result).

2.2.5 Obtaining A Tighter Chebyshev's Bound By Estimating the Variance σ^2

Consider a random variable X with the underlying distribution P_X taken from a compact interval of real numbers, i.e., $X \in [a, b]$. For N realizations $x_1, x_2, ..., x_N$, the empirical mean is given by:

$$\hat{E}_{P_X}[X] = \hat{\mu}_N = \frac{1}{N}\sum_{j=1}^{N} x_j.$$

The true mean is given by:

$$E_{P_X}[X] = \mu$$

while the true variance is given by:

$$\sigma^2 = E_{P_X}\Big[\big(X - E_{P_X}[X]\big)^2\Big].$$

The maximum variance for a random variable over $[a, b]$ is when half of the mass is at a and the other half is at b, i.e., $\sigma^2_{\text{MAX}} = R_X^2/4$ where $R_X = b - a$. Applying Chebyshev's Theorem gives us the following:

$$P(|\hat{\mu}_N - \mu| \geq \epsilon) \leq \frac{R_X^2}{4N \cdot \epsilon^2}.$$

We want to find out if a tighter upper bound for Chebyshev's bound (for any random variable distributed over a compact interval) can be obtained. If we knew $\mu = E_{P_X}[X]$, we could obtain an unbiased estimate of σ^2:

$$\hat{\sigma}^2_N = \frac{1}{N}\sum_{j=1}^{N}(x_j - \mu)^2 = \frac{1}{N}\sum_{j=1}^{N} y_j,$$

where $y_j = (x_j - \mu)^2$, $j = 1, ..., N$, are realizations of a new random variable $Y = (X - E_{P_X}[X])^2$. $\hat{\sigma}^2_N$ is the empirical mean of Y based on a sample $\{y_j\}_{j=1}^N$.

The true mean of Y is:

$$E_{P_Y}[Y] = E_{P_X}\Big[(X - E_{P_X}[X])^2\Big] = \sigma^2.$$

We can apply Chebyshev's Theorem for Y and obtain the following inequality:

$$P(|\hat{\sigma}^2_N - \sigma^2| \geq \delta) \leq \frac{\text{Var}_{P_Y}[Y]}{N \cdot \delta^2},$$

where $\text{Var}_{P_Y}[Y]$ is the variance of Y.

For $y_j = (x_j - \mu)^2$, given that the minimal possible value is $y_{\min} = 0$ (if $x_j = \mu$), and that the maximal possible value is $y_{\max} = (b-a)^2$ (if $x_j = a$ and $\mu = b$), the range for Y is $R_Y = y_{\max} - y_{\min} = (b-a)^2 - 0 = (b-a)^2$. Again, trivially, $\text{Var}_{P_Y}[Y]$ can be bounded by:

$$\text{Var}_{\text{MAX}}[Y] \leq \frac{R_Y^2}{4} = \frac{(b-a)^4}{4} = \frac{R_X^4}{4}.$$

So,

$$P(|\hat{\sigma}^2_N - \sigma^2| \geq \delta) \leq \frac{R_X^4}{4N \cdot \delta^2}.$$

We can thus summarize the following two points. First, with probability of at least $c_1 = 1 - R_X^2/(4N\epsilon^2)$ (with respect to generating many N-tuples of observations $(x_1, ..., x_N)$), we know that $\hat{\mu}$ is no further away from μ than ϵ, i.e., $|\hat{\mu} - \mu| \leq \epsilon$). That is:

$$P(|\hat{\mu}_N - \mu| \leq \epsilon) \geq 1 - \frac{R_X^2}{4N \cdot \epsilon^2}.$$

Second, with probability of at least $c_2 = 1 - R_X^4/(4N\delta^2)$, we know that $\hat{\sigma}^2_N$ is no further away from σ^2 than δ, i.e., $|\hat{\sigma}^2_N - \sigma^2| \leq \delta$). That is:

$$P(|\hat{\sigma}_N^2 - \sigma^2| \leq \delta) \geq 1 - \frac{R_X^4}{4N \cdot \delta^2}.$$

We would like to use $\hat{\sigma}_N^2 + \delta$ as an upper bound for σ^2 (when σ^2 is lower than the maximum possible value $R_X^4/4$). However, we have assumed that we know μ. In actual situations, we do not know μ.

We examine the correction required for the fact that $\hat{\mu}$ rather than μ is used to calculate the variance $\hat{\sigma}_N^2$. That is:

$$\hat{\hat{\sigma}}_N^2 = \frac{1}{N}\sum_{j=1}^{N}(x_j - \hat{\mu}_N)^2.$$

We know that with probability at least $c_1 = 1 - R_X^2/(4N\epsilon^2)$, $|\hat{\mu} - \mu| \leq \epsilon$. So, the worst case is when $\hat{\hat{\sigma}}_N^2$ underestimates $\hat{\sigma}_N^2$ (we are interested in the upper bound for σ^2) because $\hat{\mu}_N$ is closer to x_j than μ. There are two situations where we get this: (a) when $x_j < \hat{\mu}_N < \mu$ (in which case we calculate $(\hat{\mu}_N + \epsilon - x_j)^2$), and (b) when $\mu < \hat{\mu}_N < x_j$ (in which case we calculate $(x_j - \hat{\mu}_N - \epsilon)^2$).

As such, with probability (with respect to generation of many N-tuples $(x_1, ..., x_N)$ from P_X^N) $c_1 = 1 - R_X^2/(4N\epsilon^2)$, we have $(x_j - \hat{\mu}_N - \epsilon)^2 \geq (x_j - \mu)^2$. So, with probability c_1, we can be sure that:

$$\hat{\sigma}_{N,U}^2 = \frac{1}{N}\sum_{j=1}^{N}(x_j - \hat{\mu}_N - \epsilon)^2 \geq \hat{\sigma}_N^2.$$

With probability $c_2 = 1 - R_X^4/(4N\delta^2)$, we have the upper bound $\sigma^2 \leq \hat{\sigma}_N^2 + \delta$.

In order to combine the probabilities in a simple way, we must make the events independent. So, we need two sets of N-tuples of observations. The first set of $x_{1,1}, x_{1,2}, ..., x_{1,N}$ is used to estimate $\hat{\mu}_{1,N} = \frac{1}{N}\sum_{j=1}^{N} x_{1,j}$. The second set $x_{2,1}, x_{2,2}, ..., x_{2,N}$ is used to estimate $\hat{\mu}_{2,N} = \frac{1}{N}\sum_{j=1}^{N} x_{2,j}$ and $\hat{\sigma}_{N,U}^2 = \frac{1}{N}\sum_{j=1}^{N}(x_{2,j} - \hat{\mu}_{2,N} - \epsilon)^2$.

Then, with probability:

$$c_1 \cdot c_2 = \Big(1 - \frac{R_X^2}{4N \cdot \epsilon^2}\Big) \cdot \Big(1 - \frac{R_X^4}{4N \cdot \delta^2}\Big),$$

we have $\sigma^2 \leq \hat{\sigma}_{N,U}^2 + \delta$ since we know that $\sigma^2 \leq \hat{\sigma}_N^2 + \delta$ and $\hat{\sigma}_N^2 \leq \hat{\sigma}_{N,U}^2$.

As such, one can claim with probability at least $c_1 c_2$ that the following inequality holds:

$$P(|\hat{\mu}_{1,N} - \mu| \geq \epsilon) \leq \frac{\hat{\sigma}_{N,U}^2 + \delta}{N \cdot \epsilon^2}.$$

However, for this inequality to be of use, we require that $\sigma^2 \leq \hat{\sigma}_{N,U}^2 + \delta < \sigma_{\text{MAX}}^2 = R_X^2/4$.

We apply this result to obtain a tighter upper bound for the generalization framework introduced earlier, which we present as Lemma 2.

Lemma 2 *For a strategy i, consider two independent non-overlapping sets of N test strategies: T_1 and T_2, where $T_1 \cap T_2 = \emptyset$ and $|T_1| = |T_2| = N$. The first set is used to estimate the generalization performance $\hat{G}_i(T_1) = \frac{1}{N}\sum_{j \in T_1} G_i(j)$. The second set is used to estimate the variance $\hat{\sigma}_{N,U}^2 = \frac{1}{N}\sum_{j \in T_2}\big(G_i(j) - \hat{G}_i(T_2) - \epsilon\big)^2$, for some positive number $\epsilon > 0$, where $\hat{G}_i(T_2) = \frac{1}{N}\sum_{j \in T_2} G_i(j)$. Then, for $\delta > 0$ with probability at least $c_1 c_2 = (1 - R^2/(4N\epsilon^2))(1 - R^4/(4N\delta^2))$, the following inequality holds:*

$$P_N(|\hat{G}_i(T_1) - G_i| \geq \epsilon) \leq \frac{\hat{\sigma}^2_{N,U} + \delta}{N \cdot \epsilon^2}. \quad (6)$$

3 Examples of Chebyshev's Bound for Estimating Generalization Performance in IPD Games

We have addressed the difficulty for problems where the true generalization performance cannot be determined by solving analytically a closed-form formula and is computationally prohibitive by estimating the generalization performance using a random sample of test strategies. However, a large sample of N random test strategies may be required to claim with a high probability for a small precision value that the estimated value is close to the true value of the generalization performance. It is known that the Chebyshev's bound given earlier is a loose upper bound.

Here, we conduct an empirical study for some practical games-the IPD games-and show that one can expect to obtain a better estimate of the generalization performance compared to the theoretical value given by Chebyshev's bound when considering the worst-case (i.e., actual $P(|D_N|' \geq \epsilon')$ is lower than the Chebyshev's bound). We also show how a strategy's performance profile with respect to the strategy space, the variance σ^2, can be used through a second estimation to obtain a tighter Chebyshev's bound. Finally, we show that games can affect the accuracy of the estimation procedure because they affect the strategy performance profile with respect to the strategy space.

3.1 The IPD Game

In the IPD game, two players engaged in repeated interactions are given two choices, cooperate and defect [47, 48, 49]. The dilemma embodies the tension between rationality of individuals who are tempted to defect and rationality of the group where every individual is rewarded by mutual cooperation [49] since both players who are better off mutually cooperating than mutually defecting are vulnerable to exploitation by one of the party who defects.

The classical IPD game can be formulated by considering a predefined payoff matrix that specifies the payoff that a player receives for the choice it makes given the choice that the opponent makes. Referring to the payoff matrix given by Figure 2, both players receive R (*reward*) units of payoff if both cooperate. They both receive P (*punishment*) units of payoff if they both defect. However, when one player cooperates while the other defects, the cooperator will receive S (*sucker*) units of payoff while the defector receives T (*temptation*) units of payoff. With the classic IPD game, the values R, S, T, and P must satisfy the constraints: $T > R > P > S$ and $R > (S + T)/2$. Any set of values can be used as long as they satisfy the IPD constraints (we use $T = 5$, $R = 4$, $P = 1$, and $S = 0$ in all our experiments unless stated otherwise). The game is played when both players choose between the two alternative choices over a series of moves (i.e., repeated interactions).

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

Figure 2: The payoff matrix framework of a two-player, two-choice game. The payoff given in the lower left-hand corner is assigned to the player (row) choosing the move, while that of the upper right-hand corner is assigned to the opponent (column).

The classical IPD game has been extended to more complex versions of the game to better model real-world interactions. One simple example that we will use in this paper is the IPD with multiple, discrete levels of cooperation [21, 25, 26, 50, 51]. The n-choice IPD game is defined by the payoffs obtained through the following linear interpolation:

$$p_{\mathrm{A}} = 2.5 - 0.5c_{\mathrm{A}} + 2c_{\mathrm{B}}, \quad -1 \leq c_{\mathrm{A}}, c_{\mathrm{B}} \leq 1, \tag{7}$$

where p_{A} is the payoff to player A, given that c_{A} and c_{B} are the cooperation levels of the choices that players A and B make, respectively. For example, for the *three*-choice and *four*-choice IPD games, the payoff matrix is given by Figures 3 and 4 [25], respectively.

		PLAYER B		
		$+1$	0	-1
PLAYER A	$+1$	4	2	0
	0	$4\frac{1}{2}$	$2\frac{1}{2}$	$\frac{1}{2}$
	-1	5	3	1

Figure 3: The payoff matrices for the two-player *three*-choice IPD game. Each element of the matrix gives the payoff for Player A.

		PLAYER B			
		$+1$	$+\frac{1}{3}$	$-\frac{1}{3}$	-1
PLAYER A	$+1$	4	$2\frac{2}{3}$	$1\frac{1}{3}$	0
	$+\frac{1}{3}$	$4\frac{1}{3}$	3	$1\frac{2}{3}$	$\frac{1}{3}$
	$-\frac{1}{3}$	$4\frac{2}{3}$	$3\frac{1}{3}$	2	$\frac{2}{3}$
	-1	5	$3\frac{2}{3}$	$2\frac{1}{3}$	1

Figure 4: The payoff matrices for the two-player *four*-choice IPD game [25]. Each element of the matrix gives the payoff for Player A.

Note that in generating the payoff matrix for an n-choice IPD game, the following conditions must be satisfied [25]:

1. For $c_{\mathrm{A}} < c'_{\mathrm{A}}$ and constant c_{B}: $p_{\mathrm{A}}(c_{\mathrm{A}}, c_{\mathrm{B}}) > p_{\mathrm{A}}(c'_{\mathrm{A}}, c_{\mathrm{B}})$,
2. For $c_{\mathrm{A}} \leq c'_{\mathrm{A}}$ and $c_{\mathrm{B}} < c'_{\mathrm{B}}$: $p_{\mathrm{A}}(c_{\mathrm{A}}, c_{\mathrm{B}}) < p_{\mathrm{A}}(c'_{\mathrm{A}}, c'_{\mathrm{B}})$, and
3. For $c_{\mathrm{A}} < c'_{\mathrm{A}}$ and $c_{\mathrm{B}} < c'_{\mathrm{B}}$: $p_{\mathrm{A}}(c'_{\mathrm{A}}, c'_{\mathrm{B}}) > (1/2)(p_{\mathrm{A}}(c_{\mathrm{A}}, c'_{\mathrm{B}}) + p_{\mathrm{A}}(c'_{\mathrm{A}}, c_{\mathrm{B}}))$.

These conditions are analogous to those for the classical IPD's. The first condition ensures that defection always pays more. The second condition ensures that mutual cooperation has a higher payoff than mutual defection. The third condition ensures that alternating between cooperation and defection does not pay in comparison to just playing cooperation.

In the IPD, defection is not necessarily the best choice of play. Many studies have shown cooperative play to be a viable strategy [48, 49]. More importantly, cooperative play can be learned from an initial, random population through co-evolutionary learning [2, 52, 53, 54, 55], which is different from the classical evolutionary games approach that considers frequency dependent reproductions of predetermined strategies (e.g., "ecological approach" [49, 56]) that do not involve an adaptation process on some strategy representations.

With the learning of IPD strategies, one important question is whether the learned IPD strategy is "robust". Axelrod [2] considered robustness of a strategy in the context of its performance against some expert-designed strategies. Darwen and Yao [33, 34, 35] defined robustness in the context of generalization. These previous studies [2, 33, 34, 35] approached the issue of robustness (i.e., generalization) in an empirical and *ad hoc* manner. Here, we investigate the issue by applying the generalization framework introduced earlier.

3.2 Different Definitions of Generalization Performance

Before applying the generalization framework to the co-evolutionary learning of a game, we first need to define specifically the generalization performance so that when we say that a strategy generalizes well, it is according to some well-defined performance criteria (which is determined by the definition of game outcomes). For zero-sum games like chess, the game outcome for a strategy can be easily defined, e.g., win, lose, or draw[3]. When we apply the generalization framework with this definition of game outcome for chess, we refer to how well co-evolutionary learning can produce a strategy that can win against as many strategies as possible.

However, for the nonzero-sum game such as IPD, there is no specific notion of "win" that one defines for a game outcome. Instead, the "best" strategy is usually the one with the highest average payoffs of some N games played (if one includes each strategy also playing against its own twin), i.e., round robin tournament. In other words, the game outcome is either the average payoff per move, or the total payoff. This is not the same as the case when one sums the number of individual "wins", where a win is defined as having higher average payoff per move in a game.

Here, we explore the generalization performance of co-evolutionary learning for the IPD games from two perspectives, i.e., we consider two different definitions of generalization performance. This is to allow a more in-depth investigation as to how co-evolutionary learning produces strategies in terms of their generalization, i.e., whether the strategy generalizes to a specific performance criterion, or to more than one performance criterion.

3.2.1 Generalization Performance in Terms of the Number of Wins Based on Individual Outcomes

First, we define generalization performance in terms of the number of wins based on individual outcomes of games (e.g., a game between a pair of strategies). This definition of generalization performance follows the notion from zero-sum games where one is interested in the performance of winning against as many known strategies as possible. If $g(i, j)$ refers directly to the average payoff per move to strategy i when it plays an IPD game with strategy j, then a game outcome, which can either be win or lose, is defined as follows:

$$G_{\mathrm{W}}(i,j) = \begin{cases} C_{\mathrm{WIN}} & for \quad g(i,j) > g(j,i), \\ C_{\mathrm{LOSE}} & for \quad otherwise, \end{cases}$$

where C_{WIN} and C_{LOSE} are constants that can be arbitrarily specified as long as $C_{\mathrm{WIN}} > C_{\mathrm{LOSE}}$[4].

[3] This is the widely accepted convention, although for our purpose here of determining generalization performance, we can just consider a case of win and lose, i.e., a draw is considered a lose.

[4] In this paper, we use $C_{\mathrm{WIN}} = G_{\mathrm{MAX}} = 100$ and $C_{\mathrm{LOSE}} = G_{\mathrm{MIN}} = 0$. We arbitrarily choose these values for presentation of results.

Using Equation 3, the estimated generalization performance based on a sample of N test strategies is given by:

$$\hat{G}_{\mathrm{W}}(i) = \frac{1}{N} \sum_{j \in S_N} G_{\mathrm{W}}(i,j). \tag{8}$$

We use $G(i) = G_i$ to make the notations simpler. We denote $G_{\mathrm{W}}(i)$ to represent the true generalization performance for the definition introduced above.

3.2.2 Generalization Performance in Terms of Average Payoff

Second, we define generalization performance in terms of average payoff. This definition of generalization performance follows the notion of average performance against a population of strategies (tournaments for IPD games). Here, the game outcome is the average payoff per move, i.e., $G_{\mathrm{A}}(i,j) = g(i,j)$, where $G_{\mathrm{MAX}} = T$ and $G_{\mathrm{MIN}} = S$ can be determined from Equation 7. As such, the generalization performance refers to the average payoff from a round robin tournament of games. However, for simplicity, and since N is a large number and assuming there is always a $j = i$ in S_N:

$$\hat{G}_{\mathrm{A}}(i) = \frac{1}{N} \sum_{j \in S_N} G_{\mathrm{A}}(i,j), \tag{9}$$

so that it is straightforward to use the Chebyshev's Theorem to obtain the Chebyshev's bound. We denote $G_{\mathrm{A}}(i)$ to represent the true generalization performance for the definition introduced above.

3.3 Chebyshev's Bounds in IPD Games

For the two definitions of generalization performance, Chebyshev's Theorem can be applied to obtain their respective Chebyshev's bounds. However, as mentioned earlier, the Chebyshev's bound is an upper bound in probability, and in most practical situations, σ^2 is less than $R^2/4$. That is, even though we are limited by using a sample of random test strategies of size N in estimating the generalization performance, we expect that for a particular precision value ϵ', the probability $P(|D_N|' \geq \epsilon')$ is lower than the Chebyshev's bound. Here, we investigate this empirically.

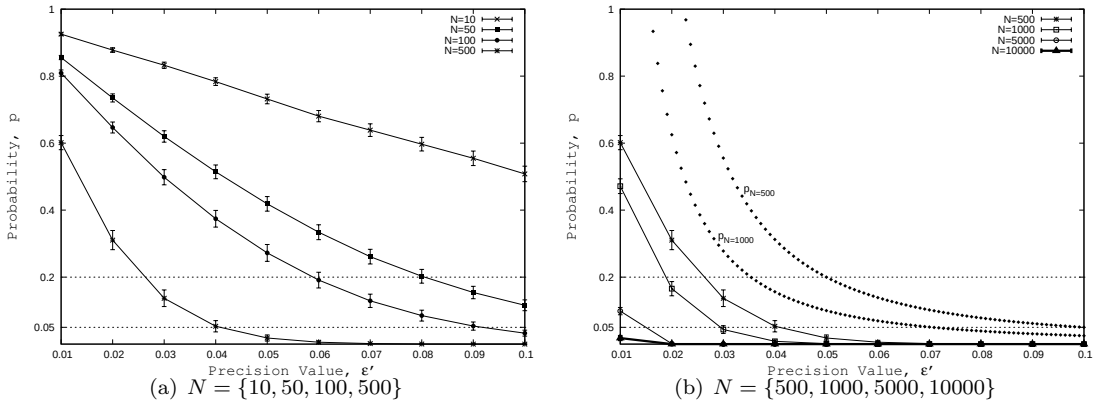
For simplicity, we will restrict our investigation to the case of the IPD game with deterministic and reactive memory-one strategies that consider their previous moves and opponent's previous moves. We consider the *three*-choice IPD game. We assume that strategies are uniformly distributed in the strategy space $\mathbf{S}$. For this case, it is possible to compute the true generalization performance since the strategy space $\mathbf{S}$ of $M = 59049$ unique strategies is not too big. It is not possible for the *four*-choice IPD game, while it is trivial for the *two*-choice IPD game with $M = 32$.

3.3.1 Chebyshev's Bounds for Different Generalization Performance in IPD Games

Experiments were conducted for the two definitions of generalization performance introduced earlier. For all the experiments, the estimated and true generalization performance were determined for 4000 strategies that were randomly sampled. The proportion of the 4000 strategies where $|D_N|' > \epsilon'$ was determined. This proportion can be considered as a rough estimate of the actual probability $P(|D_N|' \geq \epsilon')$. Experiments were repeated for 50 samples of random test strategies S_N for different Ns (rough % of M) at 10 (0.02%), 50 (0.1%), 100 (0.2%), 500 (1%), 1000 (2%), 5000 (10%), 10000 (20%) that are drawn independently[5].

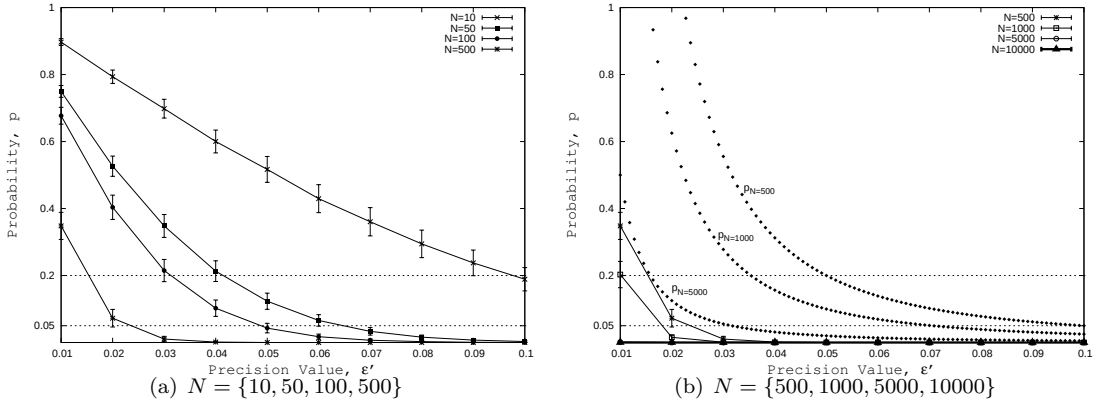
Figure 5 shows the results of the experiments for the generalization performance defined by $G_{\mathrm{W}}(i)$ and estimated with $\hat{G}_{\mathrm{W}}(i)$. From the figure, it can be observed that the probability $P(|D_N|' \geq \epsilon')$ (roughly

[5]We have verified for each sample S_N that no strategy is obtained more than once.



(a) $N = \{10, 50, 100, 500\}$ (b) $N = \{500, 1000, 5000, 10000\}$

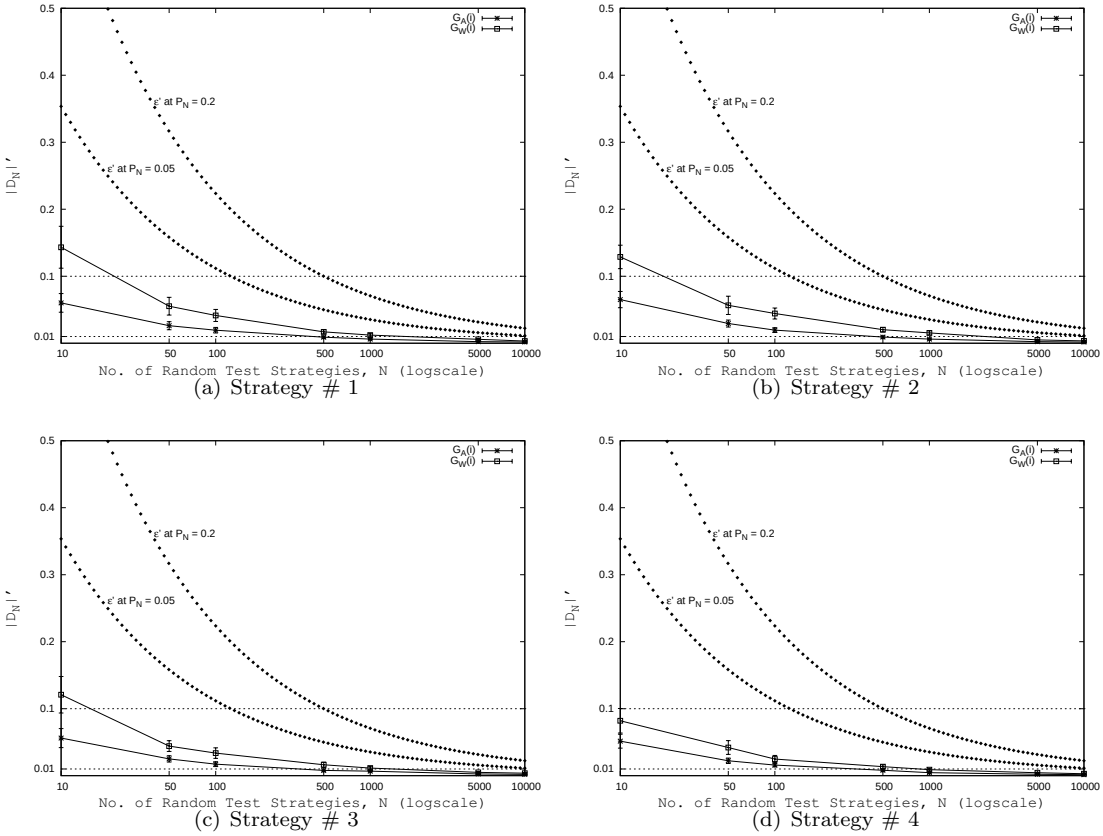
Figure 5: Figures showing the probability that $|D_N|' > \epsilon'$ for ϵ' in $[0.01, 0.1]$ where the sample of random test strategies S_N of size N is used to compute $\hat{G}_{\mathrm{W}}(i)$. Each graph is obtained by averaging from results of using 50 independent samples S_N with error bars representing 95% confidence interval. P_N gives the curves for the Chebyshev's bounds with different Ns.



(a) $N = \{10, 50, 100, 500\}$ (b) $N = \{500, 1000, 5000, 10000\}$

Figure 6: Figures showing the probability that $|D_N|' > \epsilon'$ for ϵ' in $[0.01, 0.1]$ where the sample of random test strategies S_N of size N is used to compute $\hat{G}_{\mathrm{A}}(i)$. Each graph is obtained by averaging from results of using 50 independent samples S_N with error bars representing 95% confidence interval. P_N gives the curves for the Chebyshev's bounds with different Ns.

estimated by the proportion of 4000 samples of $|\hat{G}_{\mathrm{W}}(i) - G_{\mathrm{W}}(i)|/R$, $i = 1, ..., 4000$) reduces as N increases from 10 to 10000 for ϵ' in $[0.01, 0.1]$. The empirical probability curves are observed to be lower than those given by Chebyshev's bounds. Similar observation is also made for the generalization performance defined by $G_{\mathrm{A}}(i)$ and estimated with $\hat{G}_{\mathrm{A}}(i)$ in Figure 6.



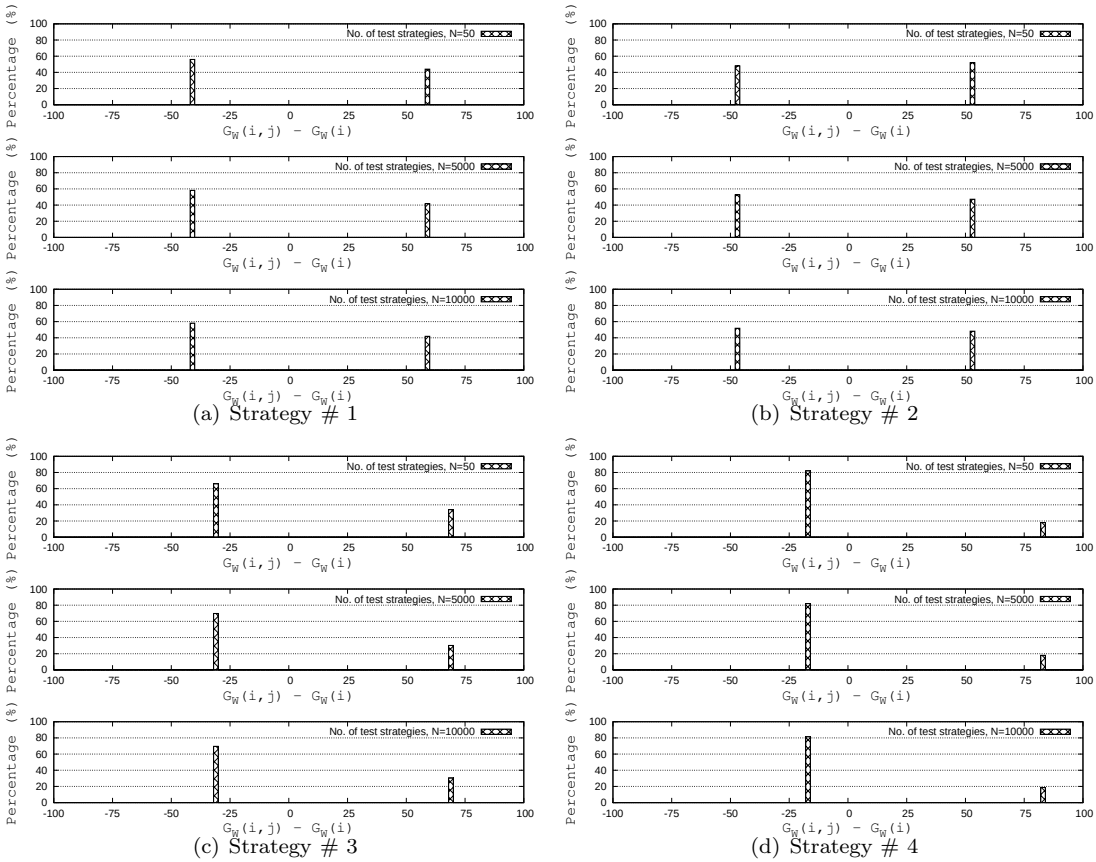
(a) Strategy # 1 (b) Strategy # 2

(c) Strategy # 3 (d) Strategy # 4

Figure 7: (a)-(d) shows for four different strategies i, the normalized absolute differemce, $|D_N|'$, is obtained for each definition of generalization performance (e.g., $G_{\mathrm{W}}(i)$ and $G_{\mathrm{A}}(i)$). Each graph is obtain by averaging from results of using 50 independent samples S_N with error bars representing 95% confidence interval. Each subfigure also contains the curves of precision value $\epsilon' = 1/\sqrt{4NP_N}$ obtained from Chebyshev's bounds that are labelled according the probability P_N used.

Experiments show that normalized $|\hat{G}_{\mathrm{W}}(i) - G_{\mathrm{W}}(i)|$ is usually larger for all Ns compared to normalized $|\hat{G}_{\mathrm{A}}(i) - G_{\mathrm{A}}(i)|$ (e.g., Fig. 7). This can be explained by analyzing the distribution (histogram) of $G(i,j) - G(i)$ for individual strategies (that we randomly picked) against a sample of test strategies (that we also randomly picked from many samples). Figures 8 and 9 show for four strategies, the distribution of $G_{\mathrm{W}}(i,j) - G_{\mathrm{W}}(i)$ and $G_{\mathrm{A}}(i,j) - G_{\mathrm{A}}(i)$, respectively, for a particular sample of S_N for N at 50, 5000, and 10000. On the one hand, the distribution of $G_{\mathrm{W}}(i,j) - G_{\mathrm{W}}(i)$ is always bimodal, i.e., having two separate peaks, which may lead to higher σ^2. On the other hand, the distribution of $G_{\mathrm{A}}(i,j) - G_{\mathrm{A}}(i)$ spreads over an interval, leading to lower σ^2. Considering the Chebyshev's bound from Equation 6, it is more likely $|\hat{G}_{\mathrm{W}}(i) - G_{\mathrm{W}}(i)| \geq \epsilon$ for higher values of ϵ when σ^2 (which can be estimated with respect to S_N) is higher.

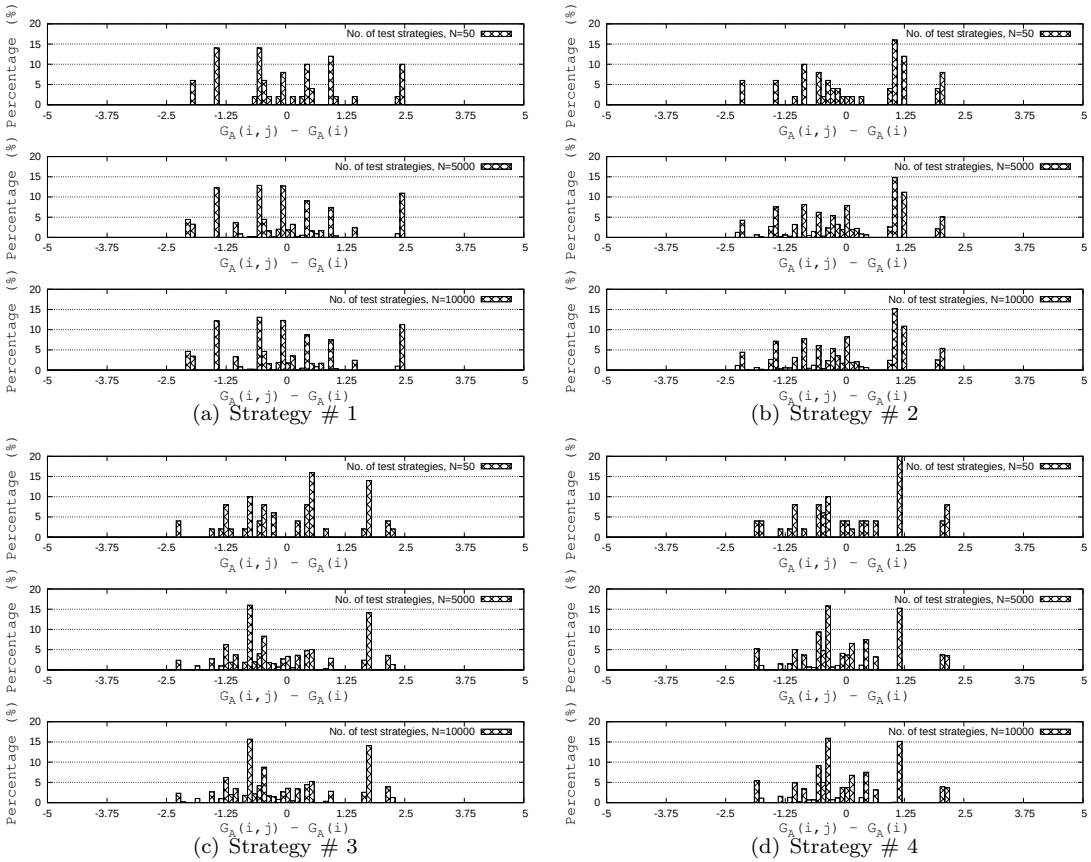
In addition, another observation is that the distribution of $G(i,j) - G(i)$ for the two definitions of generalization performance is quite similar for different sizes of S_N starting from a small N of 50 (Figs. 8 and 9). These results indicate two implications for the estimation of generalization performance for IPD games investigated here. First, the estimation from using a smaller S_N is as accurate as the estimation from using larger S_Ns. Second, the estimations are stable in terms of varying sample sizes starting from a small S_N.



(a) Strategy # 1 (b) Strategy # 2

(c) Strategy # 3 (d) Strategy # 4

Figure 8: (a) - (d) shows for four different strategies i, the distribution of $G_{\mathrm{W}}(i,j) - G_{\mathrm{W}}(i)$ for N at 50, 5000 and 10000.



(a) Strategy # 1 (b) Strategy # 2

(c) Strategy # 3 (d) Strategy # 4

Figure 9: (a) - (d) shows for four different strategies i, the distribution of $G_{\mathrm{A}}(i,j) - G_{\mathrm{A}}(i)$ for N at 50, 5000 and 10000.

3.3.2 Can We Find A Tighter Chebyshev's Bound?

The Chebyshev's bound given by Equation 5 assumes the worst-case and takes $\sigma^2_{\text{MAX}} = R^2/4$ to upper-bound the probability. We have earlier provided a mathematical analysis to find a tighter bound that requires the estimation of σ^2 so that rather than using $\sigma^2_{\text{MAX}} > \sigma^2$, $\hat{\sigma}^2_{N,U} + \delta$ is used instead (with some probability $c_1 c_2$ that can be determined), i.e., a tighter Chebyshev's bound given by Equation 6.

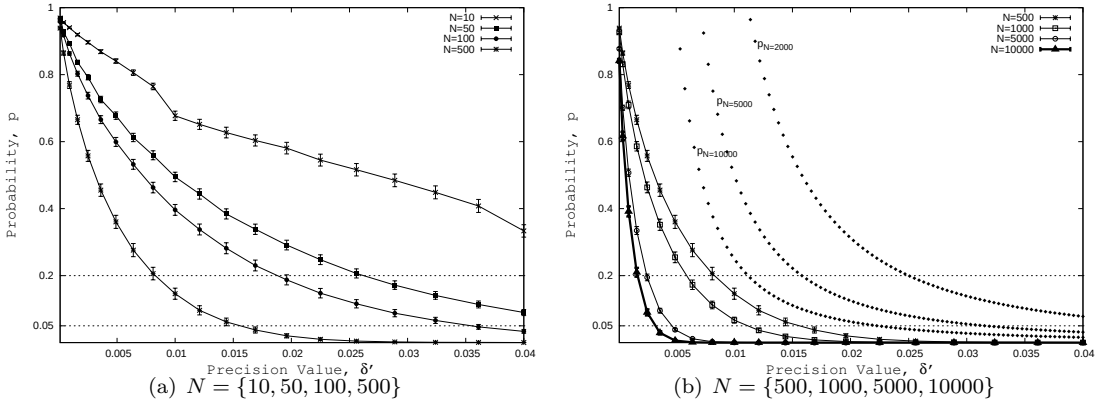
Note that to have a high probability $c_1 c_2$ where $c_1 = 1 - R^2/(4N\epsilon^2)$ and $c_2 = 1 - R^4/(4N\delta^2)$, we want c_2 to be as high as possible by having a higher value for δ (which is where this tighter bound is useful because we are hoping that σ^2 is small and that $\hat{\sigma}^2_{N,U} + \delta < \sigma^2_{\text{MAX}}$ even for a larger value of δ).

We apply Chebyshev's Theorem for the random variable $|\hat{\sigma}^2_N - \sigma^2|$ to investigate empirically how good is the estimation of σ^2 when using a sample of test strategies for the *three*-choice IPD game (we can actually compute the exact value of σ^2 for this game):

$$P(|\hat{\sigma}^2_N - \sigma^2| \geq \delta) \leq \frac{R^4}{4N \cdot \delta^2}.$$

We can simplify the above by letting $|D(\hat{\sigma}^2_N)|' = |\hat{\sigma}^2_N - \sigma^2|/R^2$ and $\delta' = \delta/R^2$ so that we now obtain:

$$P(|D(\hat{\sigma}^2_N)|' \geq \delta') \leq \frac{1}{4N \cdot \delta'^2}.$$



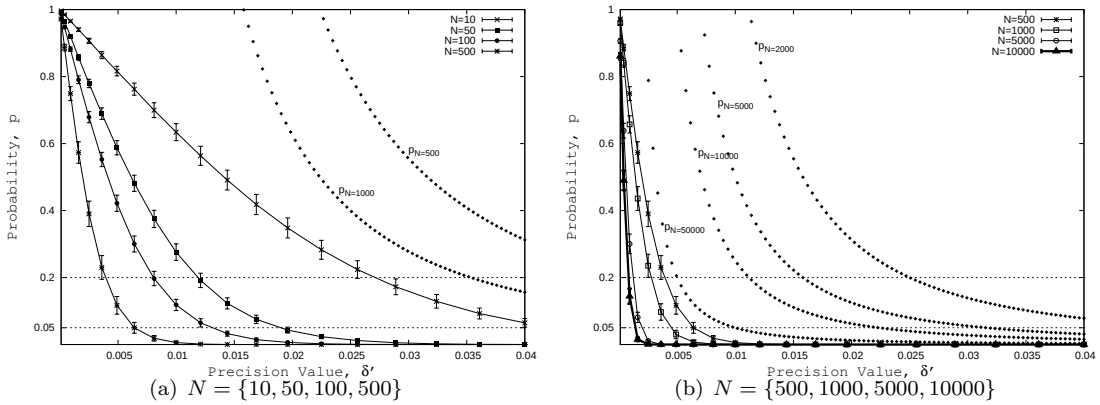
(a) $N = \{10, 50, 100, 500\}$ (b) $N = \{500, 1000, 5000, 10000\}$

Figure 10: Figures showing the probability that $|D(\hat{\sigma}^2_N)|' > \delta'$ for δ' in $[0.0001, 0.04]$ where the sample of random test strategies S_N of size N is used to compute $\hat{\sigma}^2_N$ for $G_{\text{W}}(i)$. Each graph is obtained by averaging from results of using 50 independent samples S_N with error bars representing 95% confidence interval. P_N gives the curves for the Chebyshev's bounds with different Ns.

Experiments were conducted for the two definitions of generalization performance to determine the probability (proportion) of 4000 random strategies where $|D(\hat{\sigma}^2_N)|' > \delta'$ for 50 samples of S_N from 10 to 10000 that are drawn uniformly and independently[6]. Figures 10 and 11 summarize the results, which show that generally the empirical probability curves are lower for estimation of σ^2 for $G_{\text{A}}(i)$ compared to the case of $G_{\text{W}}(i)$. More importantly, the results show that a high probability can be claim that $|D(\hat{\sigma}^2_N)|' \leq \delta'$ if we consider a large precision value of δ'.

We also present results for the estimation of σ^2 for individual strategies. Table 1 shows the $\hat{\sigma}^2_N$ for various N and also the true value σ^2 for some random strategies for $G_{\text{W}}(i)$. Table 2 shows the results of $|\hat{\sigma}^2_N - \sigma^2|$ for $G_{\text{W}}(i)$. If we consider a large precision value of $\delta' = 0.04$ ($\delta = 400$), we know from Chebyshev's bound for example that for $N = 10000$, the probability that $|\hat{\sigma}^2_N - \sigma^2| \leq \delta$ is $1 - 1/(4 \times 10000 \times 0.04^2) > 0.98$.

[6]Note that we actually compute $\hat{\hat{\sigma}}^2_N$ rather than $\hat{\sigma}^2_N$. Regardless however, the following results will show that most of the time, $\hat{\hat{\sigma}}^2_N$ overestimates σ^2. Furthermore, at the end of the section, we will show results of corrected estimates $\hat{\sigma}^2_{N,U}$ and how it can be used to obtain a tighter Chebyshev's bound.



(a) $N = \{10, 50, 100, 500\}$ (b) $N = \{500, 1000, 5000, 10000\}$

Figure 11: Figures showing the probability that $|D(\hat{\sigma}^2_N)|' > \delta'$ for δ' in $[0.0001, 0.04]$ where the sample of random test strategies S_N of size N is used to compute $\hat{\sigma}^2_N$ for $G_{\mathrm{A}}(i)$. Each graph is obtained by averaging from results of using 50 independent samples S_N with error bars representing 95% confidence interval. P_N gives the curves for the Chebyshev's bounds with different Ns.

However, as indicated earlier in Figure 10(b), the actual estimation of σ^2 is better than the theoretical value given by Chebyshev's bound. For example, $|\hat{\sigma}^2_N - \sigma^2|$ for $N = 10000$ is much smaller than 400 for a probability greater than 0.98 (Table 2). Similar observations can be made for $G_{\mathrm{A}}(i)$ (Tables 3 and 4)[7].

Table 1: Comparison of $\hat{\sigma}^2_N$ for various N with the true value σ^2 for 10 strategies for $G_{\mathrm{W}}(i)$ where $\sigma^2 \in [0, 2500]$.

N	# 1	# 2	# 3	# 4	# 5	# 6	# 7	# 8	# 9	# 10
10	2500	2400	2500	2400	2400	2100	0	2400	2500	1600
50	2464	2496	2244	1476	2496	1924	1204	2356	2464	2484
100	2304	2484	2059	1344	2464	1924	1275	2436	2496	2379
500	2436	2500	2224	1539	2491	1895	1317	2477	2497	2375
1000	2419	2498	2131	1470	2484	1875	1218	2482	2500	2278
5000	2432	2492	2108	1481	2490	1849	1341	2476	2499	2220
10000	2434	2497	2118	1515	2493	1816	1321	2473	2500	2231
TRUE	**2436**	**2496**	**2126**	**1475**	**2491**	**1816**	**1293**	**2477**	**2499**	**2228**

Table 2: Comparison of $|\hat{\sigma}^2_N - \sigma^2|$ for various N for 10 strategies for $G_{\mathrm{W}}(i)$ where $\sigma^2 \in [0, 2500]$.

N	# 1	# 2	# 3	# 4	# 5	# 6	# 7	# 8	# 9	# 10
10	64	96	374	925	91	284	1293	77	1	628
50	28	0	118	1	5	108	89	121	35	256
100	132	12	67	131	27	108	18	41	3	151
500	0	4	98	64	0	79	24	0	2	147
1000	17	2	5	5	7	59	75	5	1	50
5000	4	4	18	6	1	33	48	1	0	8
10000	2	1	8	40	2	0	28	4	1	3

A tighter Chebyshev's bound can be found if $\hat{\sigma}^2_N + \delta < \sigma^2_{\mathrm{MAX}}$. For example, for a particular strategy (# 7 in Table 1 for $G_{\mathrm{W}}(i)$), $\sigma^2 = 1293$ while $\hat{\sigma}^2_N = 1321$. The corrected estimate is found to be $\hat{\sigma}^2_{N,U} = 1337$ (where the probability that $\hat{\sigma}^2_{N,U} \geq \hat{\sigma}^2_N$ is $c_1 = 1 - 1/(4 \times 10000 \times 0.04^2) > 0.98$). In this case, we

[7]Note that results presented in the tables refer to the estimated generalization performance of a strategy i when a sample S_N of random test strategies is used. As such, $\hat{\sigma}^2_N$ in Table 1 for a strategy can fluctuate (higher or lower) around the true value σ^2. For very small sample size such as $N = 10$ that is out of Chebyshev's bound, erratic results such as $\hat{\sigma}^2_N = 0$ can be obtained. As for results in Tables 2 and 4, increasing N does not necessarily result with decreasing $|D_N|$. However, all the results are within theoretical results from Chebyshev's bounds (e.g., with a probability greater than 0.98 and using $N = 10000$ random test strategies, δ is no greater than 400 for the case of $G_{\mathrm{W}}(i)$).

Table 3: Comparison of $\hat{\sigma}_N^2$ for various N with the true value σ^2 for 10 strategies for $G_{\rm A}(i)$ where $\sigma^2 \in [0, 6.25]$.

N	# 1	# 2	# 3	# 4	# 5	# 6	# 7	# 8	# 9	# 10
10	2.06	1.63	1.77	1.80	2.48	1.22	1.72	2.27	0.99	1.11
50	1.54	1.39	1.35	1.28	2.21	0.90	1.52	2.10	1.21	0.92
100	1.43	1.32	1.24	1.31	2.23	0.95	1.53	2.04	1.40	0.88
500	1.56	1.33	1.44	1.03	2.11	0.91	1.38	2.06	1.43	0.89
1000	1.55	1.34	1.40	1.00	2.12	0.92	1.40	2.03	1.46	0.90
5000	1.55	1.32	1.34	0.96	2.11	0.88	1.43	1.93	1.47	0.86
10000	1.59	1.34	1.35	0.97	2.15	0.87	1.46	1.95	1.49	0.87
TRUE	**1.59**	**1.35**	**1.37**	**0.96**	**2.15**	**0.88**	**1.44**	**1.96**	**1.48**	**0.86**

Table 4: Comparison of $|\hat{\sigma}_N^2 - \sigma^2|$ for various N for 10 strategies for $G_{\rm A}(i)$ where $\sigma^2 \in [0, 6.25]$.

N	# 1	# 2	# 3	# 4	# 5	# 6	# 7	# 8	# 9	# 10
10	0.47	0.28	0.4	0.84	0.33	0.34	0.28	0.31	0.49	0.25
50	0.05	0.04	0.02	0.32	0.06	0.02	0.08	0.14	0.27	0.06
100	0.16	0.03	0.13	0.35	0.08	0.07	0.09	0.08	0.08	0.02
500	0.03	0.02	0.07	0.07	0.04	0.03	0.06	0.1	0.05	0.03
1000	0.04	0.01	0.03	0.04	0.03	0.04	0.04	0.07	0.02	0.04
5000	0.04	0.03	0.03	0	0.04	0	0.01	0.03	0.01	0
10000	0	0.01	0.02	0.01	0	0.01	0.02	0.01	0.01	0.01

know that $\hat{\sigma}_N^2$ overestimates σ^2. Regardless, with probability $c_1 c_2 = (1 - 1/(4 \times 10000 \times 0.04^2)) \times (1 - 1/(4 \times 10000 \times 0.04^2) > (0.98) \times (0.98) > 0.95)$, we know that $\sigma^2 \leq \hat{\sigma}_{N,U}^2 + \delta$ with $\delta = 400$. Here, we obtain $\hat{\sigma}_{N,U}^2 + \delta = 1337 + 400 < \sigma_{\rm MAX}^2 = 2500$ (Fig, 12). Figure 13 illustrates another example of a tighter Chebyshev's bound for $G_{\rm A}(i)$ where $\sigma^2 = 0.88$ (strategy # 6 in Table 3) and $\hat{\sigma}_{N,U}^2 = 0.92$ where $N = 10000$.

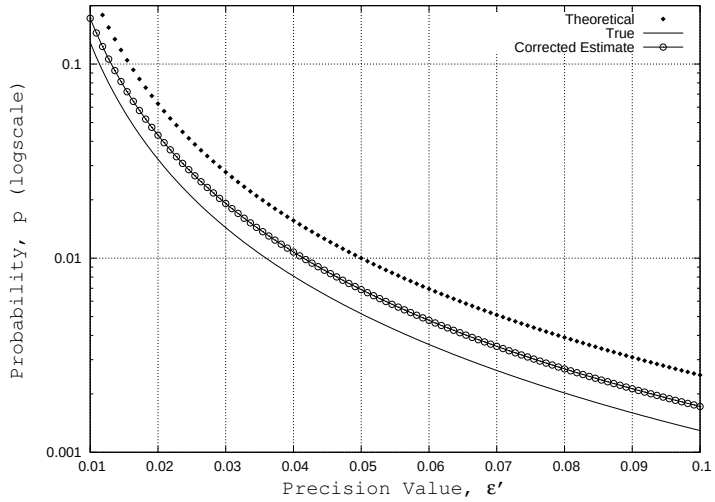


Figure 12: Example of obtaining a tighter Chebyshev's bound for $G_{\rm W}(i)$ (# 7 in Table 1). The curve "Theoretical" gives the original Chebyshev's bound if σ_{MAX}^2 is used. The curve "True" gives the actual bound if σ^2 is used. The curve "Corrected Estimate" gives the tighter Chebyshev's bound if $\hat{\sigma}_{N,U}^2 + \delta$ is used. All curves are determined for $N = 10000$.

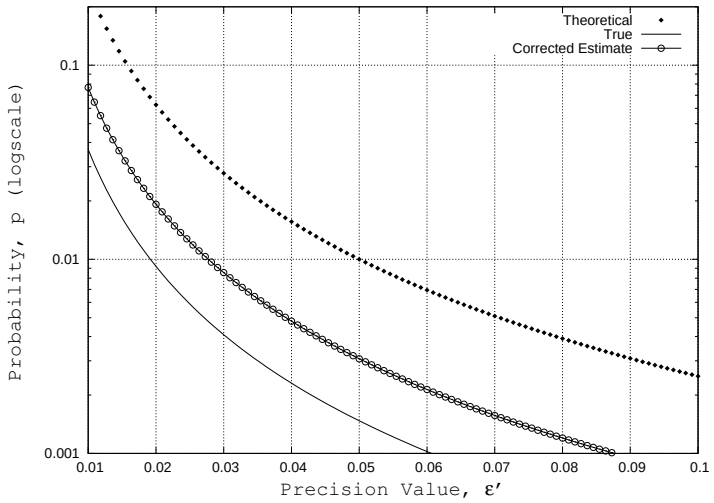


Figure 13: Example of obtaining a tighter Chebyshev's bound for $G_{\rm A}(i)$ (strategy # 6 in Table 3). The curve "Theoretical" gives the original Chebyshev's bound if σ^2_{MAX} is used. The curve "True" gives the actual bound if σ^2 is used. The curve "Corrected Estimate" gives the tighter Chebyshev's bound if $\hat{\sigma}^2_{N,U} + \delta$ is used. All curves are determined for $N = 10000$.

3.3.3 What is the Impact of Different Games on the Estimation of Generalization Performance

We know that the estimation of generalization performance of a particular strategy depends on its performance with all strategies in the strategy space (as in σ^2 following Equation 4). Furthermore, we have shown that the accuracy of the estimated of generalization performance that depends on the profile of a strategy's performance (i.e., game outcomes) against all strategies in the strategy space (e.g., σ^2 depends on the distribution of $G_i(j) - G_i$).

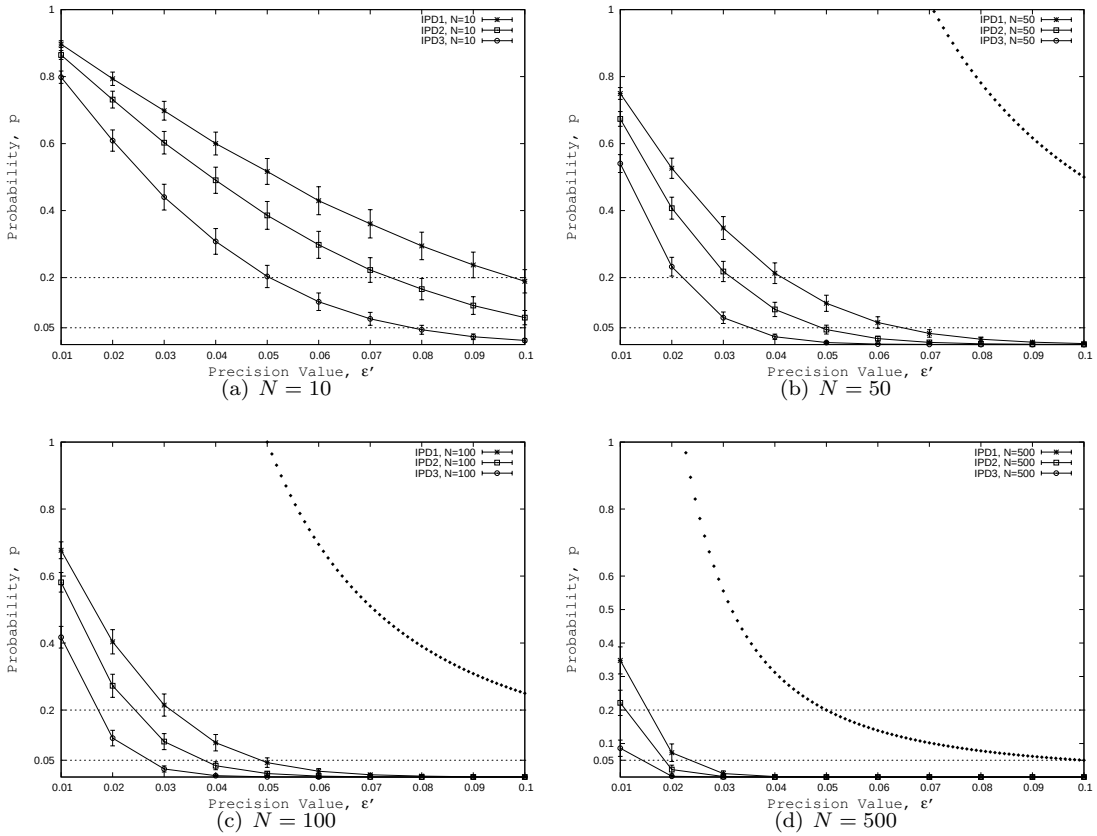
We note that the distribution of $G_i(j) - G_i$ for any strategy i depends on the game. As such, we investigate this issue with a series of experiments of IPD games with different payoff matrices: (a) IPD1 (the original game considered), (b) IPD2 (using the interpolation $p_{\rm A} = 2.5 - 0.5c_{\rm A} + 1.5c_{\rm B}$), and (c) IPD3 (using the interpolation $p_{\rm A} = 2.5 - 0.5c_{\rm A} + c_{\rm B}$).

By penalizing strategies that alternate with different choices (IPD1 penalizes the most while IPD3 penalizes the least), the games can affect the profile of a strategy's performance with respect to all strategies in the strategy space $\mathbf{S}$ given that a large proportion of strategies in $\mathbf{S}$ alternate between different choices. We repeat the earlier experiment to find the probability that $|D_N|' > \epsilon'$ for these IPD games and compare the results.

In general, the same results for $G_{\rm W}(i)$ are obtained for all the IPD games. This is because for $G_{\rm W}(i)$, we count the number of wins for each pair-wise interaction. Although the average payoff received from the pair-wise interaction changes, $G_{\rm W}(i,j)$ is the same because the relationship between $g(i,j)$ and $g(j,i)$ is still the same[8].

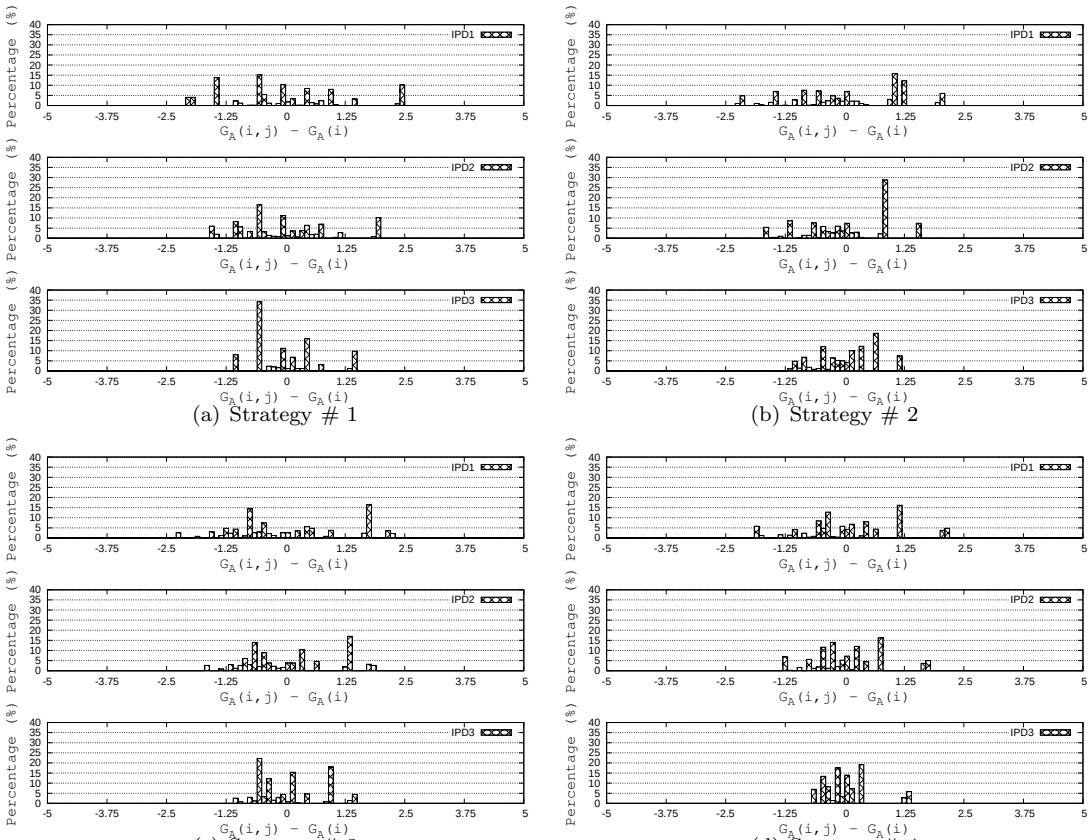
However, for $G_{\rm A}(i)$, results show for all Ns, the empirical probability curves for IPD3 is lower than IPD2, which in turn, is lower than IPD1 (Fig. 14). Note that different IPD games award payoffs for plays with alternating choices (strategies that constitute a large proportion in $\mathbf{S}$) differently. For example, the payoff differences between neighboring choices (e.g., +1 compared to 0, or 0 compared to −1) are reduced for IPD3 compared to IPD1. As such, it is not surprising that $|\hat{G}_{\rm A}(i) - G_{\rm A}(i)|$ is smaller for the same strategy i for IPD3 compared to IPD1 due to reduced payoff differences of neighboring choices. It is also observed that the distribution of $G_{\rm A}(i,j) - G_{\rm A}(i)$ for individual strategies tends to be more peaky and centers around $G_{\rm A}(i,j) - G_{\rm A}(i) = 0$ for IPD3 compared to IPD1 and IPD2 (Fig. 15). The smaller values of σ^2 for the case of IPD3 imply a more accurate estimation from Equation 4.

[8]Note also that the number of rounds at 150 is sufficiently long.



(a) $N = 10$ (b) $N = 50$ (c) $N = 100$ (d) $N = 500$

Figure 14: Comparison of the probability that $|D_N|' > \epsilon'$ for ϵ' in $[0.01, 0.1]$ where S_N of size N ((a) - (d)) is used to compute $\hat{G}_{\mathrm{A}}(i)$ for different IPD games. Each graph is obtained by averaging from results of using 50 independent samples S_N with error bars representing 95% confidence interval. P_N gives the curves for the Chebyshev's bounds with different Ns. For each subfigure where a particular value of N is used, the Chebyshev's bound for N is also plotted.



(a) Strategy # 1 (b) Strategy # 2

(c) Strategy # 3 (d) Strategy # 4

Figure 15: (a) - (d) shows for four different strategies i, the distribution of $G_{\mathrm{A}}(i,j) - G_{\mathrm{A}}(i)$ for $N = 500$ for IPD1, IPD2, and IPD3 games. Actual calculations for the variance show that the values decrease from IPD1 to IPD2 to IPD3. For example, for strategy # 1, σ_N^2 (σ^2) for IPD1, IPD2, and IPD3 are 1.56 (1.59), 0.92 (0.95), and 0.47 (0.48), respectively.

4 Estimating Generalization Performance in Co-evolutionary Learning

Here, we illustrate the application of the generalization framework to co-evolutionary learning of IPD games. In particular, we focus on simple IPD games (e.g., IPD games with *two* and *three* choices) where the true generalization performance can be computed. This allows us to investigate and show empirically that a small sample of S_N can be used to provide a good estimation of the generalization performance of co-evolutionary learning of IPD games in comparison with the prediction given by Chebyshev's bound.

4.1 Co-evolutionary Learning Model

Figure 16 illustrates the general co-evolutionary learning framework, which involves an iterative process of variations (mutation and crossover) and selection (choosing solutions for the next generation) on competing solutions that strategically interact with each other. A co-evolutionary learning system can be implemented using co-evolutionary algorithms. There are a variety of different co-evolutionary algorithms, and they can be roughly classified according to their population structure, i.e., single population co-evolution [2, 21, 25] and multiple population co-evolution (which includes two populations structure that separates solutions and tests used to evaluate fitness of solutions in their respective populations that do not mix or breed [12, 13, 38]). For simplicity, we apply the generalization framework to co-evolutionary learning with the single population structure. However, we note that the generalization framework does not depend on the learning systems, and as such, can be applied to co-evolutionary learning with the multiple population structure as well.

1. Initialize the population, X(t=1)
2. Evaluate the fitness of each individual through a comparison process with other individuals in X(t)
3. Select parents from X(t) based on their evaluated fitness
4. Generate offsprings from parents to produce X(t+1)
5. Repeat steps (2-4) until some termination criteria are met

Figure 16: The general framework of co-evolutionary learning.

4.1.1 Strategy Representation

Several strategy representations have been investigated in the past, e.g., a look-up table with bit-string encoding [2], finite state machines [52, 53, 54], and neural networks [25, 51, 57, 58]. Here, we consider the simple approach of direct look-up table strategy representation that we introduced and studied earlier in [25] that directly represents IPD strategy behaviors, a one-to-one mapping between the genotype space (strategy representation) and the phenotype space (behaviors). However, the main advantage of using the direct look-up table representation is that the search space (given by the strategy representation) and the strategy space is the same (assuming uniform strategy distribution in the strategy space $\mathbf{S}$). This simplifies and allows direct investigation on the generalization performance of co-evolutionary learning.

Figure 17 illustrates the direct look-up table representation for the strategies with two choices and memory-one [25]. $m_{ij}, i, j = 1, ..., n$ specifies the choice to be made, given the inputs i (player's own previous choice) and j (opponent's previous choice). Rather than using pre-game inputs (two for memory-one strategies), the first move is specified independently, m_{fm}. For example, for the *two*-choice IPD (Fig. 17), each table element can take any of the two choices, e.g., $+1$ and -1. For the *three*-choice IPD (Fig. 18), each table element can take $+1$, 0, and -1. Note that values in the table elements are the same as those used to produce the payoffs in the payoff matrix through a linear interpolation.

		Opponent's Previous Move	
		+1	−1
Player's	+1	m_{11}	m_{12}
Previous Move	−1	m_{21}	m_{22}

Figure 17: The direct look-up table representation for the deterministic and reactive memory-one IPD strategy that consider two choices (also includes m_{fm} for the first move, which is not shown in the figure).

		Opponent's Previous Move		
		+1	0	−1
Player's	+1	m_{11}	m_{12}	m_{13}
Previous	0	m_{21}	m_{22}	m_{23}
Move	−1	m_{31}	m_{32}	m_{33}

Figure 18: The direct look-up table representation for the deterministic and reactive memory-one IPD strategy that consider three choices (also includes m_{fm} for the first move, which is not shown in the figure).

A simple mutation operator is used to generate offspring from parents [25]. For the n-choice IPD, mutation replaces the original choice of a particular element in the direct look-up table (includes m_{fm} and m_{ij}) with one of the other remaining $n-1$ possible choices with an equal probability of $1/(n-1)$. Each table element has a fixed probability, p_{m}, of being replaced by one of the remaining $n-1$ choices. The value p_{m} is not optimized.

Crossover is not used because with the direct look-up table for strategy representation, any variation operator will introduce variations on behaviors directly. As investigated earlier in [25] (even for the more complex IPD game with intermediate choices), a simple mutation operator is more than sufficient to introduce the required variations of strategy behaviors. The use of crossover is not necessary.

4.1.2 Co-evolutionary Learning Procedure

We refer to the following co-evolutionary learning procedure [25] as the classical co-evolutionary learning (CCL):

1. Generation step, $t = 1$:
   Initialize POPSIZE/2 parent strategies, $P_i, i = 1, 2, ..., \text{POPSIZE}/2$, randomly.
2. Generate POPSIZE/2 offspring, $O_i, i = 1, 2, ..., \text{POPSIZE}/2$, from POPSIZE/2 parents using a variation.
3. All pairs of strategies compete, including the pair where a strategy plays itself (i.e., round-robin tournament). For POPSIZE strategies in a population, every strategy competes a total of POPSIZE games.
4. Select the best POPSIZE/2 strategies based on total payoffs of all games played. Increment generation step, $t = t + 1$.
5. Step 2 to 4 are repeated until termination criterion (i.e., a fixed number of generation) is met.

All IPD games involve a fixed game length of 150 iterations. A fixed and sufficiently long duration for the evolutionary process (e.g., $t = 300$) is used. Note that unlike studies such as [25] that consider the population evolving persistently to a particular behavior (e.g., mutual cooperation), here we observe

the generalization performance of co-evolutionary learning during the evolutionary process. Further note that unlike in [25], we do not consider noisy IPD games. All experiments are repeated for 30 independent runs.

4.2 Measuring Generalization Performance of Co-evolutionary Learning

The generalization framework was earlier presented in the context of a complete solution. However, co-evolutionary learning can be broadly classified either as *cooperative co-evolutionary learning* [59], where solutions represent different components that are combined to produce the complete solution, or *competitive co-evolutionary learning* whereby a solution is complete [38]. Here, we investigate generalization performance of competitive co-evolutionary learning systems such as CCL for simplicity since we can use a strategy to represent the population in terms of generalization performance. However, there is no prerequisite to choose the best evolved strategy of the population to be the representative. The best U evolved strategies from the population, $\text{SPOP}_U = \{\text{spop}_1, ..., \text{spop}_U\}$ (ordered according to their internal fitness values), can be considered, which allows the following measurements of generalization performance of co-evolutionary learning:

- Best,
$$\text{Best}(G_{\text{SPOP}_U}) = \hat{G}_{\text{spop}_1}. \tag{10}$$
- Average,
$$\text{Avg}(G_{\text{SPOP}_U}) = \frac{1}{U}\left(\sum_{l}^{U} \hat{G}_{\text{spop}_l}\right). \tag{11}$$
- Ensemble,
$$\text{Ens}(G_{\text{SPOP}_U}) = \frac{1}{N}\left(\sum_{j \in S_N} \min\Big(\sum_{l}^{U} G_{\text{spop}_l}(j), G_{\text{MAX}}\Big)\right). \tag{12}$$

In particular, "ensemble" allows the measurement of the generalization performance of the co-evolved population rather than the best strategy. That is, it can be used to determine whether the population as a whole generalizes better (i.e., a higher level of competence for the game) compared to an individual strategy. For simplicity, we assume an ensemble system with a perfect gating mechanism [36], i.e., it perfectly chooses the best evolved strategy in SPOP_U that best performs against each of the test strategies, for the "ensemble" measurement. For example, for five test strategies, if spop_1 outperforms the first two test strategies, and if spop_2 outperforms the last three test strategies, the "ensemble" of the two evolved strategies outperforms all test strategies, i.e., it will choose spop_1 for the first two test strategies, and spop_2 for the others. Note that this measurement is only relevant for $G_{\text{W}}(i)$.

4.3 Generalization Performance of Co-evolutionary Learning for Problems with Small Strategy Space

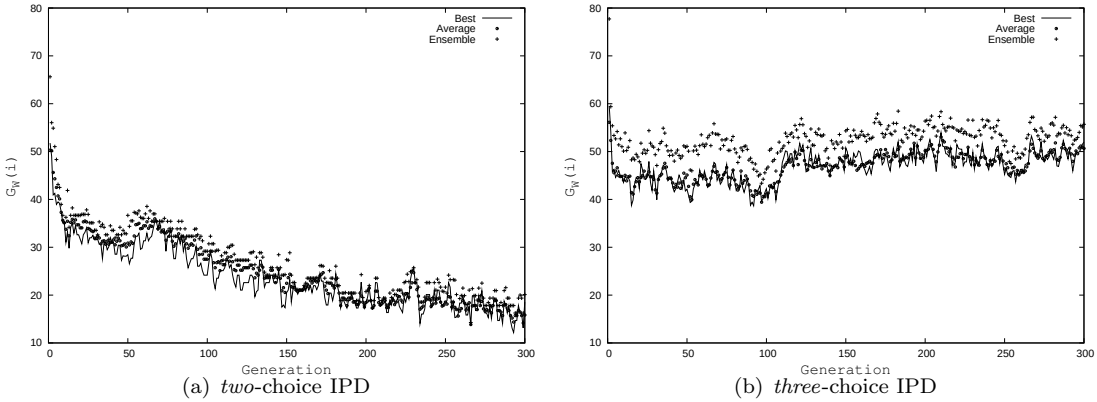
For problems with a small strategy space[9], it is possible to determine the true generalization performance of the co-evolutionary learning system using Equation 2. For example, the true generalization performance of CCL for the *two*-choice IPD and *three*-choice IPD games can be computed for $G_{\text{W}}(i)$ and $G_{\text{A}}(i)$.

For the following experiments, we used POPSIZE = 20 and 300 generations of evolutionary process for the CCL when applied to the *two*-choice and *three*-choice IPD games (note that for both IPD games, POPSIZE $< M$). Instead of taking the entire population, we considered only the top best strategies of the population (i.e., $U = 3$) when taking the $\text{Avg}(G_{\text{SPOP}_U})$ and $\text{Ens}(G_{\text{SPOP}_U})$ measurements of generalization performance.

[9] Again this should not be confused with the search space, which depends on the strategy representation. A strategy space can be small and discrete, but the search space can be infinitely large and continuous, i.e., real-valued neural network representation of a deterministic and reactive memory-one IPD strategy.

4.3.1 True Generalization Performance in the Co-evolutionary Learning of the IPD with *Two* and *Three* Choices

Figure 19 and Tables 5 and 6 summarize the generalization performance of CCL for $G_{\mathrm{W}}(i)$. Figure 20 and Tables 7 and 8 summarize for $G_{\mathrm{A}}(i)$. In general, the generalization performance of CCL during the evolutionary process differs for the *two*-choice and *three*-choice IPD games. On the one hand, for CCL of the *two*-choice IPD, generalization performance decreases as evolutionary process continues (Figs. 19(a) and 20(a), and Tables 5 and 7). On the other hand, for CCL of the *three*-choice IPD, generalization performance remains around similar levels as the evolutionary process continues (Figs. 19(b) and 20(b), and Tables 6 and 8).



(a) *two*-choice IPD (b) *three*-choice IPD

Figure 19: Generalization performance of CCL defined by $G_{\mathrm{W}}(i)$. The graph "Best" plots the generalization performance measurement given by $\mathrm{Best}(G_{\mathrm{SPOP}_U})$. The graph "Average" plots the generalization performance measurement given by $\mathrm{Avg}(G_{\mathrm{SPOP}_U})$. The graph "Ensemble" plots the generalization performance measurement given by $\mathrm{Ens}(G_{\mathrm{SPOP}_U})$. All graphs are averaged from measurements over 30 independent runs.

Table 5: The following gives the mean and error (95% confidence interval) over 30 independent runs for each measurement for generalization performance defined by $G_{\mathrm{W}}(i)$ at the start and at the end of CCL for the *two*-choice IPD.

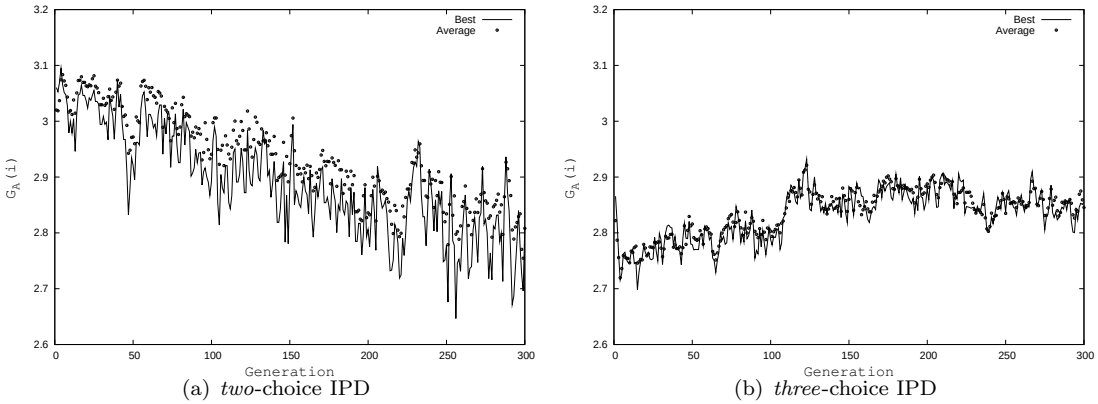
Generation	$\mathrm{Best}(G_{\mathrm{SPOP}_U})$	$\mathrm{Avg}(G_{\mathrm{SPOP}_U})$	$\mathrm{Ens}(G_{\mathrm{SPOP}_U})$
0	51.77 ± 6.32	50.28 ± 6.36	65.63 ± 6.41
300	18.54 ± 9.23	15.83 ± 7.86	20.10 ± 9.33

Table 6: The following gives the mean and error (95% confidence interval) over 30 independent runs for each measurement for generalization performance defined by $G_{\mathrm{W}}(i)$ at the start and at the end of CCL for the *three*-choice IPD.

Generation	$\mathrm{Best}(G_{\mathrm{SPOP}_U})$	$\mathrm{Avg}(G_{\mathrm{SPOP}_U})$	$\mathrm{Ens}(G_{\mathrm{SPOP}_U})$
0	59.54 ± 7.52	56.12 ± 4.99	77.73 ± 5.21
300	51.98 ± 9.14	50.66 ± 8.79	55.69 ± 8.24

Table 7: The following gives the mean and error (95% confidence interval) over 30 independent runs for each measurement for generalization performance defined by $G_{\mathrm{A}}(i)$ at the start and at the end of CCL for the *two*-choice IPD.

Generation	$\mathrm{Best}(G_{\mathrm{SPOP}_U})$	$\mathrm{Avg}(G_{\mathrm{SPOP}_U})$
0	3.06 ± 0.05	3.02 ± 0.04
300	2.81 ± 0.14	2.81 ± 0.13



(a) *two*-choice IPD (b) *three*-choice IPD

Figure 20: Generalization performance of CCL defined by $G_{\mathrm{A}}(i)$. The graph "Best" plots the generalization performance measurement given by $\mathrm{Best}(G_{\mathrm{SPOP}_U})$. The graph "Average" plots the generalization performance measurement given by $\mathrm{Avg}(G_{\mathrm{SPOP}_U})$. All graphs are averaged from measurements over 30 independent runs.

Table 8: The following gives the mean and error (95% confidence interval) over 30 independent runs for each measurement for generalization performance defined by $G_{\mathrm{A}}(i)$ at the start and at the end of CCL for the *three*-choice IPD.

Generation	$\mathrm{Best}(G_{\mathrm{SPOP}_U})$	$\mathrm{Avg}(G_{\mathrm{SPOP}_U})$
0	2.87 ± 0.06	2.82 ± 0.05
300	2.85 ± 0.10	2.85 ± 0.10

The poor generalization performance for the simpler *two*-choice IPD case is due to the population of CCL overspecializing to more naive cooperators, e.g., "all cooperate". In particular, for $G_{\mathrm{W}}(i)$, naive cooperators do not generalize well against all strategies in the strategy space because they are easily exploited. That is, for a naive cooperator, strategy i, playing against strategies j, its average payoff per move is lower or the same as that of the opponents', i.e., $g(i,j) \leq g(j,i)$ for all j in $\mathbf{S}$. Since for $G_{\mathrm{W}}(i)$, the game outcome only register a "win" for strategy i if $g(i,j) > g(j,i)$, a naive cooperator will have poor generalization performance of $G_{\mathrm{W}}(i)$. This explains why CCL generalizes poorly for $G_{\mathrm{W}}(i)$ (Fig. 19(a)) rather than $G_{\mathrm{A}}(i)$ (Fig. 20(a)).

The higher tendency of CCL to overspecialize to naive cooperators for IPD games with less choices can be explained by considering that in the search space, the proportion of naive cooperators is higher when there are less choices to play. For example, for a strategy to play "all cooperate" in the *two*-choice IPD, it must play full cooperation for the first move, i.e., $m_{\mathrm{fm}} = +1$, and play full cooperation regardless of the choices for its previous move and opponent's previous move, i.e., the following values in the direct look-up table:

$$\begin{pmatrix} +1 & +1 \\ * & * \end{pmatrix}$$

where $*$ can be $+1$ or -1 in the *two*-choice IPD. If $m_{\mathrm{fm}} = +1$ and that the choices in the first row are all $+1$, then regardless of the choices in the remaining element of the table, this strategy will always play full cooperation since it will never access elements other than those in the first row. Given that there are 2^2 combinations of such tables, one can easily calculate the proportion of "all cooperate" in $\mathbf{S}$ as $2^2/2^5$ (since the total number of unique combinations of the table is 2^{2^2+1}).

For the *three*-choice IPD case, strategies play "all cooperate" if $m_{\mathrm{fm}} = +1$ and that the table is given as follows:

$$\begin{pmatrix} +1 & +1 & +1 \\ * & * & * \\ * & * & * \end{pmatrix}$$

where $*$ can be $+1$, 0, or -1 in the *three*-choice IPD. The proportion of "all cooperate" in $\mathbf{S}$ is $3^6/3^{10}$ (since the total number of unique combinations of the table is 3^{3^2+1}). As such, one can easily obtain the proportion of "all cooperate" for the n-choice IPD, which is given by $n^{(n^2-n)}/n^{(n^2+1)} = 1/n^{(n+1)}$. It can be observed that the proportion of "all cooperate" decreases exponentially with n. For the *two*-choice IPD, the proportion is 12.5%. For the *three*-choice IPD, the proportion is approximately 1%.

It is known from our previous study of CCL for the n-choice IPD games using the same direct look-up table representation [25], the population is more likely to evolve to play mutual cooperation. However, for a population of mutual cooperators, CCL cannot distinguish between different strategies that are mutually cooperating because they have similar fitness values calculated from average payoffs from round-robin tournaments. For the *two*-choice IPD, naive cooperators constitute a large 12.5% of the search space, and as such, it is more likely for the population to drift to naive cooperators when CCL fails to distinguish between mutually cooperating strategies. This is reflected in our results that show on average, around 12% of the population is "all cooperate" for the *two*-choice IPD case compared to around 0.2% for the *three*-choice IPD case.

This result also highlights why a simple co-evolutionary learning system does not guarantee searching for a solution that generalizes well even for simple problems. This is despite the possibility of having searched for a total number of strategies during the entire evolutionary process that exceeds that of the strategy space, e.g., CCL for the *two*-choice IPD with a strategy space of 32 unique strategies. What is important to note is that in a simple co-evolutionary learning system, the selection process used is not necessarily biased towards the generalization performance considered, e.g., requiring solutions to be tested against all test cases to rank them properly at every generation.

4.3.2 Estimated Generalization Performance in the Co-evolutionary Learning of the IPD with *Three* Choices

Here, we investigate the estimated generalization performance of co-evolutionary learning using a sample of random test strategies drawn from the strategy space. We consider the *three*-choice IPD with a strategy space with $M = 59049$ unique strategies (it is trivial to investigate for the *two*-choice IPD with only $M = 32$ unique strategies). We use S_N with varying N for different Ns (rough % of M) at 10 (0.02%), 50 (0.1%), 100 (0.2%), 500 (1%), 1000 (2%), 5000 (10%), 10000 (20%) that are drawn uniformly and independently (e.g., we have verified that no strategy is included more than once).

Figures 21 and 22 give various statistics of the generalization performance of the CCL at the end of the evolutionary process for $G_{\mathrm{W}}(i)$ and $G_{\mathrm{A}}(i)$, respectively, where different Ns were used, for the *three*-choice IPD. There are two graphs in each figure. Graphs (a) and (b) plot $\hat{\sigma}_N$ and actual $|\hat{G}_i - G_i|$, respectively, for different Ns, averaged over 30 independent runs with 95% confidence interval.

We first consider graphs (a) in Figures 21 and 22. It was mathematically shown earlier that the probability $P(|\hat{G}_i - G_i| \geq \epsilon)$ is bounded by a value that is directly dependent on σ^2 and reciprocally dependent on N and ϵ^2. Here, we observed that for evolved strategies, $\hat{\sigma}_N$ remains similar (with some fluctuations) to a value smaller than the maximum value of $R/2$ as N increases. Results from Figures 21(b) and 22(b) show that actual $|\hat{G}_i - G_i|$ decreases as N increases. More importantly, the actual $|\hat{G}_i - G_i|$ obtained from using a S_N is lower than the precision value obtained from the Chebyshev's bound for a larger value of N. For example, the actual $|\hat{G}_i - G_i|$ obtained from using S_N with $N = 1000$ is lower than the ϵ given by Chebyshev's bound for $P_N = 0.05$ and $N = 10000$[10].

Earlier, we showed empirically that the Chebyshev's bound is a loose upper bound in probability for $P(|\hat{G}_i - G_i| \geq \epsilon)$, and that in general, one would expect to do better, e.g., obtain a more accurate estimate

[10] Further note that for this particular empirical result, lower $|\hat{G}_i - G_i|$ can be obtained for smaller number of N (e.g, Fig. 22(b) for N at 5000 and 10000). This is due to the particular S_N sample used to compute $\hat{G}_i$ from 30 independent runs (e.g., averaged over 30 $\hat{G}_i$s). For other S_Ns, results may differ.

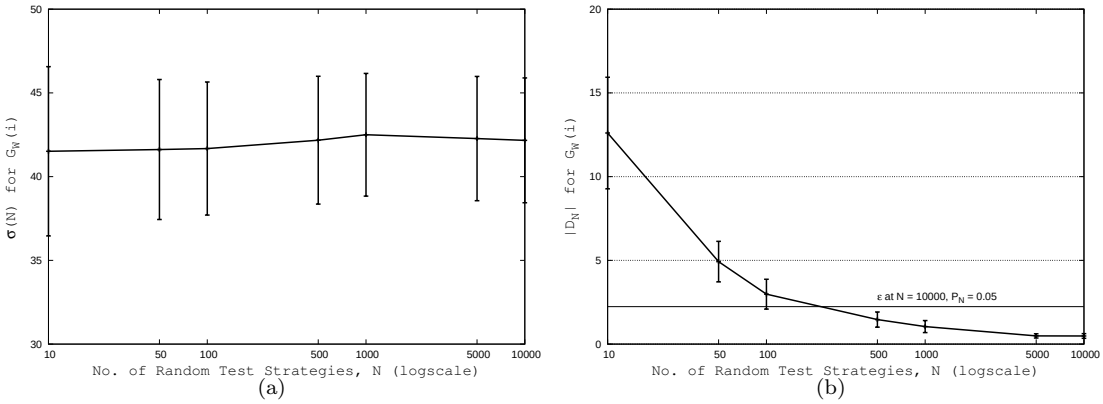


Figure 21: Statistics on generalization performance of CCL for $G_{\mathrm{W}}(i)$. (a) plots $\hat{\sigma}_N$ against N. (b) plots actual $|\hat{G}_{\mathrm{W}}(i) - G_{\mathrm{W}}(i)|$ against N. All graphs are averaged from 30 independent runs for Best(G_{SPOP_U}) measurements with error bars representing 95% confidence interval.

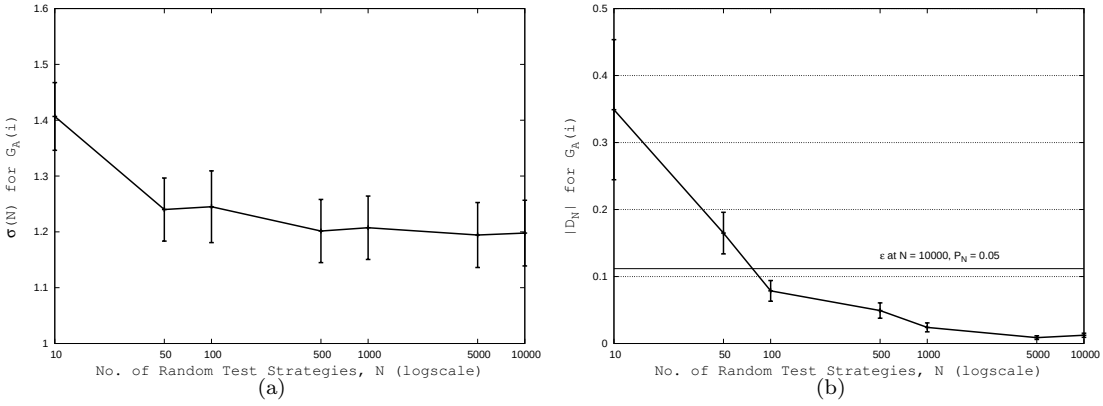


Figure 22: Statistics on generalization performance of CCL for $G_{\mathrm{A}}(i)$. (a) plots $\hat{\sigma}_N$ against N. (b) plots actual $|\hat{G}_{\mathrm{A}}(i) - G_{\mathrm{A}}(i)|$ against N. All graphs are averaged from 30 independent runs for Best(G_{SPOP_U}) measurements with error bars representing 95% confidence interval.

$\hat{G}_i$ using a sample of S_N. Here, results from actual implementation of the generalization framework to co-evolutionary learning further suggest that the estimated value $\hat{G}_i$ is closer to the true value G_i that what one can claim from Chebyshev's bound when a sample of S_N is used. In particular, for the IPD games, a small sample of random test strategies is good enough to estimate the generalization performance.

5 Estimating Generalization Performance in the Case of a Biased Test Sample

5.1 Motivation

The generalization framework provides a means to estimate the generalization performance using a random test sample. However, it is noted that the generalization performance of a solution is conditioned on the underlying distribution from which the random test sample is drawn. For different underlying distributions, the generalization performance of a solution may differ and the ranking between solutions in terms of their generalization performances may change. This raises the issue of determining which of the underlying distributions where generalization performances are measured are more important and useful given a specific problem.

For game-playing, one is more interested to estimate the generalization performance with respect to a biased test sample, e.g., biased towards test strategies that are more likely to be encountered in a real scenario such as tournaments rather than an unbiased sample where every test strategy is equally likely to be encountered. Here, one may assume that the biased test sample consists of "good" strategies based on some performance criteria, e.g., strategies that have outperformed a large number of other strategies.

5.2 Generalization Performance of Co-evolutionary Learning Based on a Biased Sample of Test Strategies

5.2.1 Obtaining A Biased Sample of Test Strategies

Many past studies have introduced and used *ad hoc* methods to obtain a biased sample. One approach is to use expert human knowledge to obtain biased test samples [2, 58]. However, there are practical limitations for this approach. First, the expert human knowledge required to obtain the biased test sample may not be readily available. Second, there is the problem of scalability in terms of competence levels that limits the utility of the biased test sample as a means from which the generalization performance is estimated, e.g., evolved strategies loosing to all test strategies.

Here, we investigate an alternative to using experts-designed test strategies with a procedure that samples the strategy space to obtain the biased test sample. It is noted that sampling for a biased test sample from the strategy space is very difficult in practice because there is no notion of a "goodness" metric in the space from which one can sample "good" strategies. Furthermore, specific properties in games such as intransitivity can present significant challenges to obtaining a useful biased test sample for the estimation of generalization performance, e.g., a strategy may be observed to generalize well simply because it exploits a specific vulnerability that all the test strategies happen to possess.

In this study, we consider a notion where a "goodness" metric can be induced on the strategy space if one can enumeratively identify and order each point (strategy) with respect to the entire space based on a performance criterion. We expect that a strategy obtained from this approach will be appropriate in cases where the game allows for some structures in the space, e.g., transitivity, where a strategy outperforms other strategies with lower performance values. A biased test sample can be obtained with a procedure that samples points of higher performances with higher probability.

However, given that this is very difficult to perform in practice most of the time (e.g., the strategy space is very large), one would have to rely on some heuristics to obtain the biased test sample. Here, we introduce a simple procedure based on the multiple partial enumerative search that extends our earlier approach in [35] to obtain a biased sample of test strategies. The procedure is described as follows:

1. Partial enumerative search run, $r = 1$:
   Sample PS strategies, $Q_i, i = 1, 2, ..., \text{PS}$, randomly.

2. Every strategy competes with all other strategies in the sample, including its twin (i.e., each strategy competes a total of PS games).

3. Detect the strategy index $s \in \{1, 2, ..., \text{PS}\}$ so that Q_s is yielding the highest total payoff. Let $Q^{(r)} := Q_s$. Increment run, $r = r + 1$.

4. Repeat steps one to three PE times to obtain PE-sized biased sample of test strategies, $Q^{(r)}, r = 1, 2, ..., \text{PE}$.

Note that the selection procedure to obtain the best test strategy is the same as the fitness evaluation used in CCL for a more direct comparison. We also note that there is intransitivy in the IPD game.

We require the population size for each partial enumerative search, PS, to be larger than the maximum number of unique strategies that can be obtained from an evolutionary run, i.e., PS > (generation × POPSIZE), for two reasons. First, the test strategy obtained from the partial enumerative search is likely not to have been encountered (trained with) by evolved strategies. Second, the test strategy have competed and outperformed a significantly larger number of opponents compared to evolved strategies, and as such, is likely to be more challenging than the opponents encountered by evolved strategies.

The procedure of repeating the partial enumerative search PE independent times is aimed at obtaining a more diverse sample of test strategies compared to the earlier approach in [35]. A sample of PE test strategies obtained from a single partial enumerative search may be less diversed as these strategies can be behaviorally similar (i.e., having the same weakness that can be exploited). Furthermore, repeating the partial enumerative search may address the problem of obtaining lower performing test strategies due to a particular poor sample of the population in a partial enumerative search.

5.2.2 Generalization Performance of Co-evolutionary Learning Based on Biased vs. Unbiased Samples for Problems with Small Strategy Space

We first compare the generalization performance of biased and unbiased samples of test strategies in the case of the *three*-choice IPD game. For the case of generalization performance with respect to biased samples of test strategies, we obtain a sample of biased strategies from 20 partial enumerative searches, which results in a sample of 20 test strategies. We investigate different samples of biased test strategies obtained from multiple partial enumerative search of different population sizes, one, four, 10, 50, 100, and 10000. Note that only when a population size of 10000 is used that the procedure satisfies the requirement of having searched more strategies compared to CCL (e.g., 10000 > generation×POPSIZE = $300 \times 20 = 6000$). Despite this, we can investigate the impact of population size used in the multiple partial enumerative search procedure in producing a biased and diverse sample of "good" test strategies that is challenging for evolved strategies.

For simplicity in notation, we refer to the estimated generalization performance using a biased sample of test strategies (e.g., $\hat{G}_i^{\text{B}}$) as $\hat{G}_{\text{W}}^{\text{B}}(i)$ and $\hat{G}_{\text{A}}^{\text{B}}(i)$ that correspond to definitions given by $G_{\text{W}}(i)$ and $G_{\text{A}}(i)$, respectively. Results from experiments for comparisons between $G_{\text{W}}(i)$ and $\hat{G}_{\text{W}}^{\text{B}}(i)$, and $G_{\text{A}}(i)$ and $\hat{G}_{\text{A}}^{\text{B}}(i)$ are summarized in Figures 23 and 24, respectively. In general, increasing the population size for the multiple partial enumerative search leads to a sample of test strategies that are "harder to defeat", e.g., evolved strategies have lower generalization performance when they competed against the sample of biased test strategies (i.e., $\hat{G}_i^{\text{B}}$) compared to the case of generalization performance against unbiased test strategies (i.e., G_i). The difference between the two generalization performances, $\hat{G}_i^{\text{B}}$ and G_i, is more pronounced especially when the biased sample of test strategies are produced using multiple partial enumerative search with larger population sizes (e.g., Figs. 23(b) and 24(b)).

Note, however, for the generalization performance defined in terms of average payoff, i.e., $\hat{G}_{\text{A}}^{\text{B}}(i)$ and $G_{\text{A}}(i)$ (Fig. 24), generalization should be viewed in the context of maximizing average payoff of oneself for any set of opponents. "Good" strategies that generalize well are those that maximize their average payoff, which does not necessarily imply minimizing the opponent's average payoff. That is, obtaining good generalization performance can be a result of cooperating with opponents rather than attempting to

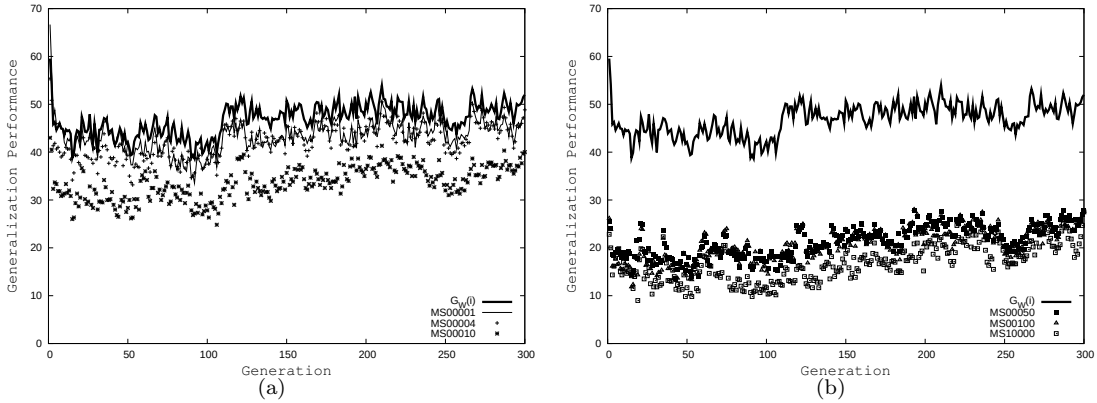


Figure 23: Comparison between $G_{\mathrm{W}}(i)$ and $\hat{G}^{\mathrm{B}}_{\mathrm{W}}(i)$s for CCL across the evolutionary process, averaged over 30 independent runs ("Best" measurements only). (a) Multiple partial enumerative search with population sizes of one, four, and ten. (b) Multiple partial enumerative search with population sizes of 50, 100, and 10000.

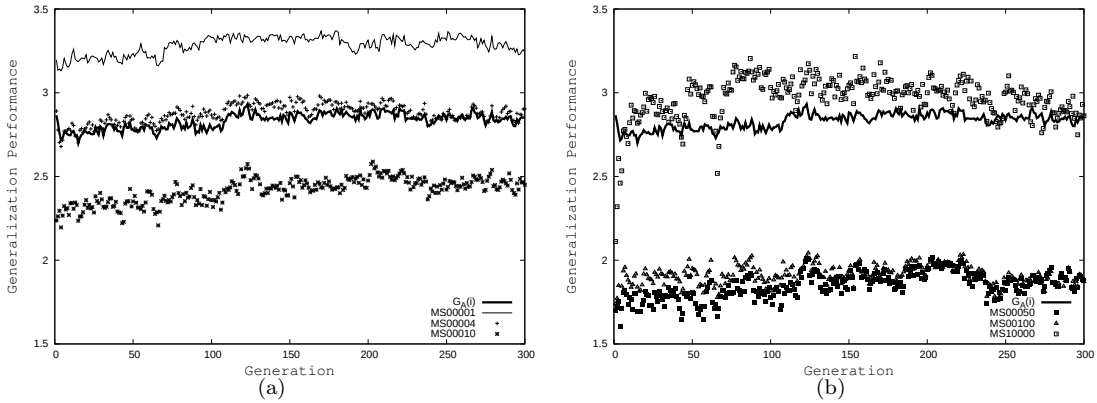


Figure 24: Comparison between $G_{\mathrm{A}}(i)$ and $\hat{G}^{\mathrm{B}}_{\mathrm{A}}(i)$s for CCL across the evolutionary process, averaged over 30 independent runs ("Best" measurements only). (a) Multiple partial enumerative search with population sizes of one, four, and ten. (b) Multiple partial enumerative search with population sizes of 50, 100, and 10000.

exploit opponents that are not naive and may retaliate. For example, it is not unexpected that results are obtained for evolved strategies where $\hat{G}_{\mathrm{A}}^{\mathrm{B}}(i)$ is higher when biased samples of test strategies obtained from the multiple partial enumerative search with increasingly higher population sizes are used. The value of $\hat{G}_{\mathrm{A}}^{\mathrm{B}}(i)$ is higher for the case of using a population size of 10000 for the multiple partial enumerative search compared to the case of using a smaller population size of 50 or 100 (Fig. 24(b)). The biased sample of test strategies obtained from the procedure with a population size of 10000 are "better" compared to those obtained from using smaller population size in the sense that they reciprocate cooperation, which results with all strategies having higher average payoffs, instead of attempting to exploit and risk other strategies retaliating, which would result with strategies having lower average payoffs.

5.2.3 Generalization Performance of Co-evolutionary Learning Based on Biased vs. Unbiased Samples for Problems with Large Strategy Space

Most practical real-world problems in game-playing often involve a very large strategy space (e.g., chess). In this case, it is not possible to compute the true generalization performance. Instead, one will have to rely on the estimated generalization performance. However, it is also possible or more likely, that for problems with very large strategy space, a large proportion of the strategies are mediocre. As such, for real-world games, one is more interested in the generalization performance of a strategy obtained from a learning system against a biased sample of "good" test strategies because such a performance comparison is more important and these strategies are more likely to be encountered in practical situations such as game tournaments.

For simplicity, instead of using complex real-world games such as chess, we investigate the *four*-choice IPD game for two reasons. First, even assuming a strategy space restricted to deterministic and reactive memory-one strategies with uniform strategy distribution in the space, it is not possible with our current computing resources to compute the true generalization performance. As such, we have to estimate the generalization performance. Second, we can easily extend the previous investigation on estimating generalization performance of co-evolutionary learning against a biased sample of test strategies obtained from the multiple partial enumerative search.

We first consider a sample of $N = 100000$ random test strategies to estimate the generalization performance against an unbiased sample. Considering $\epsilon' = 0.01$, Chebyshev's bound gives $P(|D_N|' \geq \epsilon') \leq (1/(4 \times 100000 \times 0.01^2) = 0.025$. We note that for our purpose here, a probability of at least $1 - 0.025 = 0.975 > 0.95$ that $|D_N|' \leq 0.01$ is sufficient, given the high computation cost for using S_N with $N = 100000$. However, as we have shown earlier for the *three*-choice IPD, the actual estimation may be closer to the true value compared to what one can claim with Chebyshev's bound. We conduct experiments with S_N of different sizes, e.g., N at 500, 10000, and 50000. Assuming that the estimated value obtained from using S_N with $N = 100000$ strategies is the true value, we can compare the accuracy of the estimation obtained from using smaller S_Ns. Note that for the following experiments, the CCL uses POPSIZE = 30 and that the evolutionary process last 600 generations (a moderate increase of CCL's ability to learn strategies for the more complex *four*-choice IPD game).

Table 9 summarized the results of the comparison between $|\hat{G}_i - G_i|$ for different definitions of generalization performance taken at the end of the evolutionary run of CCL with ϵ given by Chebyshev's bound when considering the probability $P_N = 0.025$ and the corresponding values of N. Note that ϵ is not adjusted because we did not calculate the true value, but instead, use an estimated value obtained with larger value of $N = 100000$. The adjusted ϵ can be higher than what is given in the table if we consider the worst-case. Results from the table shows that the actual $|\hat{G}_i - G_i|$ is significantly lower than the ϵ given by Chebyshev's bound. As such, similar to the *three*-choice IPD case, empirical results suggest that a smaller sample of S_N can be used to estimate the generalization performance and still obtain a sufficiently accurate estimation for the co-evolutionary learning of *four*-choice IPD.

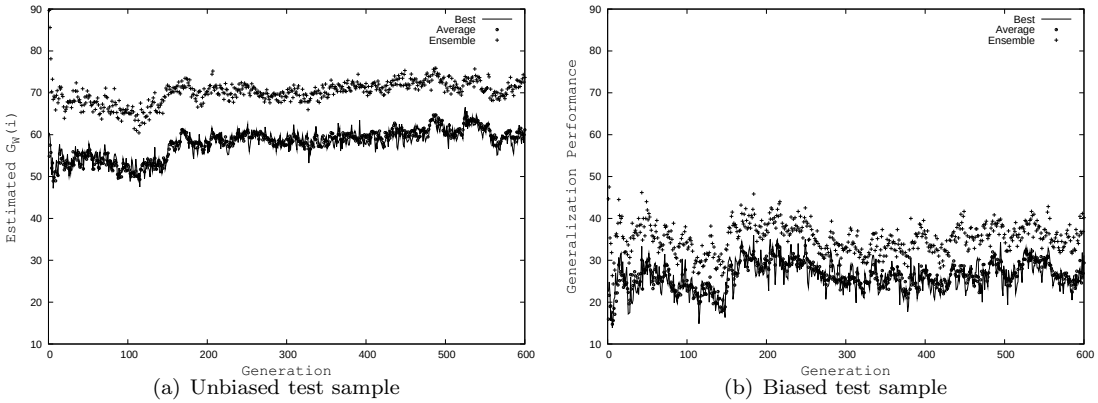
Here, we compare the generalization performance of CCL between the case of using unbiased and biased samples of test strategies for the more complex *four*-choice IPD game. The biased sample of test strategies is obtained using a multiple partial enumerative search. Each partial enumerative search is applied on a population of 10^6 strategies. This easily satisfies the requirement since CCL will search for at most, $600 \times 30 = 18000$ unique strategies. The procedure is repeated 20 times to obtain a biased sample of 20 test strategies. This allows us to investigate whether the multiple partial enumerative search procedure

Table 9: Comparison of $|\hat{G}_i - G_i|$ for different definitions of generalization performance with ϵ given by Chebyshev's bound for $P_N = 0.025$. Values for $|\hat{G}_i - G_i|$ are averaged over 30 independent runs of CCL with 95% confidence interval.

	$\hat{G}_{\mathrm{W}}(i)$		$\hat{G}_{\mathrm{A}}(i)$	
N	$\lvert\hat{G}_{\mathrm{W}}(i) - G_{\mathrm{W}}(i)\rvert$	ϵ	$\lvert\hat{G}_{\mathrm{A}}(i) - G_{\mathrm{A}}(i)\rvert$	ϵ
500	1.51 ± 0.52	20	0.040 ± 0.011	1
10000	0.48 ± 0.12	4.5	0.009 ± 0.003	0.23
50000	0.13 ± 0.04	2	0.003 ± 0.001	0.1

can produce a biased and diverse sample of test strategies that are challenging for the evolved strategies.

Results are summarized in Figures 25 and 26 for $\hat{G}_{\mathrm{W}}(i)$ and $\hat{G}^{\mathrm{B}}_{\mathrm{W}}(i)$, and $\hat{G}_{\mathrm{A}}(i)$ and $\hat{G}^{\mathrm{B}}_{\mathrm{A}}(i)$, respectively. The most important observation is that in general, comparison between $\hat{G}^{B}_{i}$ and $\hat{G}_i$ for the CCL on the more complex *four*-choice IPD is similar to that of the *three*-choice IPD case investigated earlier, i.e., evolved strategies do not perform well against a biased sample compared to an unbiased sample. For example, evolved strategies that generalized well for the search space (Fig. 25(a)) in terms of winning individual games performed poorly against the biased sample of test strategies (Fig. 25(b)). However, for the case of $\hat{G}_{\mathrm{A}}(i)$, unlike the *three*-choice IPD case, evolved strategies were unable to maximize their average payoff against the biased sample of test strategies (Figs. 26(a) and 26(b)). This result is due in part to CCL producing strategies that are less cooperative when the number of choices for the IPD game is increased since the proportion of naive cooperators in the strategy space is smaller.



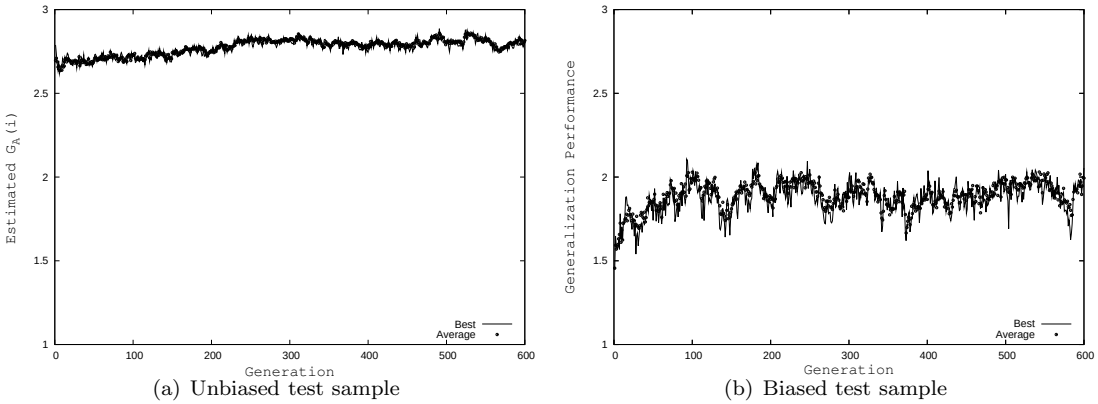
(a) Unbiased test sample (b) Biased test sample

Figure 25: Comparison between $\hat{G}_{\mathrm{W}}(i)$ and $\hat{G}^{\mathrm{B}}_{\mathrm{W}}(i)$ of CCL across the evolutionary process with "Best", "Average", and "Ensemble" measurements, for the *four*-choice IPD. All graphs are averaged over 30 independent runs.

6 Conclusion

In this paper, we have presented a theoretical framework for measuring the generalization performance of co-evolutionary learning. Although the generalization framework is presented in terms of game-playing, our result is more general and not restricted to problems of game-playing. In particular, a strategy's generalization performance is its average performance against all strategies. As such, the best generalization performance is the maximum average performance against all strategies. Although the definition is simple, it can be difficult to determine by solving analytically a closed-form formula and is computationally prohibitive.

To address this problem, we have presented a principled approach to estimate the generalization performance by computing the average performance against a sample of random test strategies. We have provided a mathematical analysis of the probability that the absolute difference between the estimated



(a) Unbiased test sample (b) Biased test sample

Figure 26: Comparison between $\hat{G}_{\mathrm{A}}(i)$ and $\hat{G}^{\mathrm{B}}_{\mathrm{A}}(i)$ of CCL across the evolutionary process with "Best" and "Average" measurements, for the *four*-choice IPD. All graphs are averaged over 30 independent runs.

and true value exceeds a given error (precision value) is bounded by a value that is reciprocally dependent only on the square of the error and the size of test sample that is used. As such, regardless of the complexity of the game (or problem in general), one can obtain a better estimate by increasing the size of the test sample. However, a tighter probability bound can be obtained for a strategy if the variance of its performance against random test strategies is known (which can be estimated as well). In addition, we showed that games also affect the accuracy of the estimation for some sizes of random test samples because they affect the variance of a strategy's performance against random test strategies drawn from the strategy space.

We have illustrated the generalization framework to determine the generalization performance of co-evolutionary learning of the n-choice IPD games. For the empirical study, we have investigated two definitions of generalization performance based on different performance criteria, e.g., in terms of the number of wins based on individual outcomes and in terms of average payoff. In particular, it is known that the bounds in probability (Chebyshev's bounds) that we used in our analysis is a loose bound, and that in general, the actual estimated value is closer to the true value than predicted by Chebyshev's bounds. We have shown experimentally for the IPD games, a small test sample can be used to provide a sufficiently accurate estimation of the generalization performance.

In the context of game-playing, the generalization in terms of average performance against "good" strategies (e.g., found in tournaments) may be more important than that against all strategies. We have introduced and investigated a simple approach using the multiple partial enumerative search to obtain a diverse sample of test strategies that are challenging to allow the estimation of generalization performance against the biased test sample. This approach does not require human expertise, and provides a more direct and meaningful comparison between evolved strategies and the test strategies because the later strategies are obtained from enumerative search of a larger population size compared to the maximum that co-evolutionary learning can achieve. We have demonstrated the approach on the co-evolutionary learning of IPD games. In particular, experiments show that the generalization performance of a co-evolutionary learning system against a biased test sample is lower compared to that of an unbiased test sample for IPD games.

The theoretical framework introduced here is the first step towards applying a rigorous and quantitative approach to analyze the performance of co-evolutionary learning through the notion of generalization from machine learning. It is now possible to analyze whether co-evolutionary learning leads to increasing generalization performance for a particular problem. For example, one can now investigate earlier claims whether diversity maintenance techniques actually help to improve generalization performance of co-evolutionary learning. The generalization framework allows for a quantitative comparison of whether a particular co-evolutionary learning system generalizes better compared to other systems. In addition, the generalization framework also provides tools for analysis and further study of co-evolutionary dynamics in context of how generalization performance changes during an evolutionary process. For example, the impact of different selection procedures to how generalization performance changes during co-evolution

can be investigated.

Acknowledgment

The authors would like to thank Dr. Simon Lucas for his role as the acting Editor-in-Chief for this paper. The authors would also like to thank the anonymous associate editor and three reviewers for their constructive comments that have helped to improve this paper significantly. The authors are grateful to The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, The University of Birmingham, UK, for providing computing support in running the experiments, and to the School of Computer Science at the University of Birmingham for its studentship to the first author. This work is partially supported by an EPSRC grant (GR/T10671/01) to the third author.

A Applying Chebyshev's Theorem

Chebyshev's Theorem [45]: *Consider a random variable U distributed according to the probability density $p(u)$. Given a positive number $a > 0$, we can bound the probability that $U \leq -a$ or $U > a$, i.e., the probability that U falls outside $[-a, +a]$, by*

$$P(|U| \geq a) \leq \frac{E[U^2]}{a^2},$$

where $E[U^2]$ is the mean of the new random variable $V = U^2$ with respect to p.

First,

$$\begin{aligned} D_N &= \hat{G}_i - G_i \\ &= \frac{1}{N} \sum_{j \in S_N} G_i(j) - G_i \\ &= \frac{1}{N} \sum_{j \in S_N} G_i(j) - \frac{1}{N} \sum_{j \in S_N} G_i \qquad \text{(since } G_i \text{ is a constant)} \\ &= \frac{1}{N} \sum_{j \in S_N} \Big(G_i(j) - G_i \Big) \\ &= \frac{1}{N} \sum_{j \in S_N} g_j \end{aligned}$$

where $g_j = G_i(j) - G_i$. Applying Chebyshev's Theorem with D_N as the random variable, we obtain the following:

$$P(|D_N| \geq \epsilon) = P(|\hat{G}_i - G_i| \geq \epsilon) \leq \frac{E_{P_N(S_N)}\big[D_N^2(S_N)\big]}{\epsilon^2}$$

where $E_{P_N(S_N)}\big[D_N^2(S_N)\big]$ is with respect to distribution $P_N = \underbrace{P_1 \times ... \times P_1}_{N \text{ times}} = P_1^N$ for a positive number $\epsilon > 0$.

Next, we need to determine $E_{P_N(S_N)}\big[D_N^2(S_N)\big] = E_{P_N(S_N)}\big[(\hat{G}_i(S_N) - G_i)^2\big]$, which is given as follows:

$$E_{P_N(S_N)}\big[D_N^2(S_N)\big] = E_{P_N(S_N)}\left[\Big(\frac{1}{N} \sum_{j \in S_N} g_j\Big) \cdot \Big(\frac{1}{N} \sum_{k \in S_N} g_k\Big)\right]$$

$$
\begin{aligned}
&= E_{P_N(S_N)}\left[\frac{1}{N^2}\left(\sum_{\substack{j,k\in S_N\\ j\neq k}} E_{P_2(j,k)}\Big[g_j\cdot g_k\Big] + \sum_{j\in S_N} E_{P_1(j)}[g_j^2]\right)\right]\\
&= \frac{1}{N^2}\sum_{j,k\in S_N} E_{P_2(j,k)}\Big[g_j\cdot g_k\Big].
\end{aligned}
$$

We note the following points:

1. We pick strategies j and k ($j\neq k$) into sample S_N *independently*, which implies
$$E_{P_2(j,k)}[g_j\cdot g_k] = E_{P_1(j)}[g_j]\cdot E_{P_1(k)}[g_k].$$
2. We use the same rules to pick j and k ($j\neq k$), which implies,
$$E_{P_1(j)}[g_j] = E_{P_1(k)}[g_k].$$
3. We can obtain the following:
$$
\begin{aligned}
E_{P_1(j)}[g_j] &= E_{P_1(j)}[G_i(j) - G_i]\\
&= E_{P_1(j)}[G_i(j)] - G_i \qquad \text{(since } G_i \text{ is a constant)}\\
&= 0 \qquad\qquad\qquad\quad \text{(since } E_{P_1(j)}[G_i(j)] = G_i\text{)}
\end{aligned}
$$
and that the same applies to $E_{P_1(k)}[g_k]$ as well.
4. As such,
$$E_{P_2(j,k)}[g_j\cdot g_k] = \begin{cases} 0, & \text{if } j\neq k,\\ E_{P_1(j)}[g_j^2], & \text{if } j = k.\end{cases}$$

Given the above, we can now obtain $E_{P_N(S_N)}\big[D_N^2(S_N)\big]$ as follows:

$$
\begin{aligned}
E_{P_N(S_N)}\big[D_N^2(S_N)\big] &= \frac{1}{N^2}\sum_{j\in S_N} E_{P_1(j)}[g_j^2]\\
&= \frac{1}{N^2}\cdot N\cdot E_{P_1(j)}[g_j^2] \qquad \text{(since } E[{g_j}^2] \text{ is a constant)}\\
&= \frac{E_{P_1(j)}[g_j^2]}{N}\\
&= \frac{\sigma_i^2}{N},
\end{aligned}
$$

where $\sigma_i^2 = E_{P_1(j)}[g_j^2]$.

References

[1] X. Yao. Introduction. *Informatica (Special Issue on Evolutionary Computation)*, 18:375–376, 1994.

[2] R. Axelrod. The evolution of strategies in the iterated prisoner's dilemma. In L. D. Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 3, pages 32–41. Morgan Kaufmann, New York, 1987.

[3] K. Chellapilla and D. B. Fogel. Evolution, neural networks, games, and intelligence. *Proceedings of the IEEE*, 87(9):1471–1496, September 1999.

[4] D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, 1994.

[5] T. Bäck, U. Hammel, and H. -P. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[6] X. Yao. *Evolutionary Computation: Theory and Applications*. World Scientific Publication, Singapore, 1999.

[7] S. Luke and R. P. Wiegand. When coevolutionary algorithms exhibit evolutionary dynamics. In A. Barry, editor, *Workshop on Understanding Coevolution: Theory and Analysis of Coevolutionary Algorithms (GECCO'02)*, pages 236–241, 2002.

[8] E. Popovici and K. De Jong. Relationships between internal and external metrics in co-evolution. In *Proc. IEEE 2005 Congress on Evolutionary Computation (CEC'05)*, pages 2800–2807, Edinburgh, UK, September 2–5, 2005.

[9] S. Nolfi and D. Floreano. Co-evolving predator and prey robots: Do "arms races" arise in artificial evolution? *Artificial Life*, 4(4):311–335, 1998.

[10] S. G. Ficici and J. B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In C. Adami, R. K. Belew, H. Kitano, and C. Taylor, editors, *Artificial Life VI: Proc. of the Sixth Int. Conf. on Artificial Life*, pages 238–247, Cambridge, MA, 1998. The MIT Press.

[11] R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In *Proc. of the 2001 Genetic and Evolutionary Computation Conference (GECCO'01)*, pages 702–709, California, USA, 2001.

[12] E. D. de Jong and J. B. Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2):159–192, 2004.

[13] J. Cartlidge and S. Bullock. Combating coevolutionary disengagement by reducing parasite virulence. *Evolutionary Computation*, 12(2):193–222, 2004.

[14] J. Noble. Finding robust texas hold'em poker strategies using pareto coevolution and deterministic crowding. In *Proc. of the 2002 International Conference on Machine Learning and Applications (ICMLA'02)*, pages 233–239, Nevada, USA, 2002.

[15] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

[16] S. Y. Chong, M. K. Tan, and J. D. White. Observing the evolution of neural networks learning to play the game of othello. *IEEE Transactions on Evolutionary Computation*, 9(3):240–251, 2005.

[17] T. P. Runarsson and S. M. Lucas. Coevolution versus self-play temporal difference learning for acquiring position evaluation in small-board go. *IEEE Transactions on Evolutionary Computation*, 9(6):540–551, 2005.

[18] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[19] H. Juille and J. B. Pollack. Co-evolving the "ideal" trainer: Application to the discovery of cellular automata rules. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Proc. of the Third Annual Conference of Genetic Programming 1998*, pages 519–527, Wisconsin, USA, 1998. Morgan Kaufmann.

[20] J. Werfel, M. Mitchell, and J. P. Crutchfield. Resource sharing and coevolution in evolving cellular automata. *IEEE Transactions on Evolutionary Computation*, 4(4):388–393, 2000.

[21] P. J. Darwen and X. Yao. Co-evolution in iterated prisoner's dilemma with intermediate levels of cooperation: Application to missile defense. *International Journal of Computational Intelligence and Applications*, 2(1):83–107, 2002.

[22] C. D. Rosin and R. K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

[23] S. G. Ficici. *Solution Concepts in Coevolutionary Algorithms*. PhD thesis, Brandeis University, Massachusetts, USA, 2004.

[24] S. G. Ficici and J. B. Pollack. Pareto optimality in coevolutionary learning. In *Advances in Artificial Life: 6th European Conference (ECAL'01)*, volume 2159 of *Lecture Notes in Computer Science*, pages 316–325. Springer, Prague, Czech Republic, September 10–14, 2001.

[25] S. Y. 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.

[26] P. J. Darwen and X. Yao. Why more choices cause less cooperation in iterated prisoner's dilemma. In *Proc. IEEE 2001 Congress on Evolutionary Computation (CEC'01)*, pages 987–994, Seoul, Korea, May 27–30, 2002.

[27] S. G. Ficici, O. Melnik, and J. B. Pollack. A game-theoretic and dynamical-systems analysis of selection methods in coevolution. *IEEE Transactions on Evolutionary Computation*, 9(6):580–602, 2005.

[28] D. Cliff and G. F. Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In *Advances in Artificial Life: Proc. of the Third European Conference on Artificial Life*, volume 929 of *Lecture Notes in Computer Science*, pages 200–218. Springer-Verlag, 1995.

[29] D. Floreano and S. Nolfi. God save the red queen! competition in co-evolutionary robotics. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proc. of the Second Annual Conference*, pages 398–406, Stanford University, CA, USA, 1997. Morgan Kaufmann.

[30] K. O. Stanley and R. Miikkulainen. The dominance tournament method of monitorihg progress in coevolution. In *Bird of a Feather Workshop: Proc. of the 2002 Genetic and Evolutionary Computation Conference (GECCO'02)*, pages 242–248, New York, USA, 2002.

[31] A. Bader-Natal and J. B. Pollack. Towards metrics and visualizations sensitive to coevolutionary failures. In *Coevolutionary and Coadaptive Systems Workshop: 2005 AAAI Symposium*, Washington DC, USA, 2005.

[32] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[33] X. Yao, Y. Liu, and P. J. Darwen. How to make best use of evolutionary learning. In R. Stocker, H. Jelinck, B. Burnota, and T. Bossomaier, editors, *Complex Systems - From Local Interactions to Global Phenomena*, pages 229–242. IOS Press, Amsterdam, 1996.

[34] P. J. Darwen and X. Yao. On evolving robust strategies for iterated prisoner's dilemma. In *Progress in Evolutionary Computation*, volume 956 of *Lecture Notes in Artificial Intelligence*, pages 276–292. Springer, 1995.

[35] P. J. Darwen. *Co-evolutionary Learning by Automatic Modularization with Speciation*. PhD thesis, University of New South Wales, Sydney, Australia, November 1996.

[36] P. J. Darwen and X. Yao. Speciation as automatic categorical modularization. *IEEE Transactions on Evolutionary Computation*, 1(2):101–108, 1997.

[37] Y. G. Seo, S. B. Cho, and X. Yao. Emergence of cooperative coalition in nipd game with localization of interaction and learning. In *Proc. IEEE 1999 Congress on Evolutionary Computation (CEC'99)*, pages 877–884, Piscataway, NJ, 1999. IEEE Press.

[38] C. D. Rosin. *Coevolutionary Search Among Adversaries*. PhD thesis, University of California, San Diego, California, USA, 1997.

[39] R. P. Wiegand and M. A. Potter. Robustness in cooperative coevolution. In *Proc. of the 2006 Conference on Genetic and Evolutionary Computation (GECCO'06)*, pages 369–376, Seattle, USA, 2006.

[40] M. Bowling. Convergence and no-regret in multiagent learning. In *Advances in Neural Information Processing Systems 17*, pages 209–216, 2005.

[41] R. Powers and Y. Shoham. Computing best-response strategies via sampling. Technical report, Computer Science Department, Stanford University, 2003.

[42] S. G. Ficici and J. B. Pollack. A game-theoretic memory mechanism for coevolution. In *Proc. of the 2003 Conference on Genetic and Evolutionary Computation (GECCO'03)*, volume 2723 of *Lecture Notes in Computer Science*, pages 286–297. Springer-Verlag, 2003.

[43] E. D de Jong. The incremental pareto-coevolution archive. In *Proc. of the 2004 Conference on Genetic and Evolutionary Computation (GECCO'04)*, pages 525–536, 2004.

[44] E. D de Jong. The maxsolve algorithm for coevolution. In *Proc. of the 2005 Conference on Genetic and Evolutionary Computation (GECCO'05)*, pages 483–489, New York, USA, 2005.

[45] B. V. Gnedenko and G. V. Gnedenko. *Theory of Probability*. Taylor & Francis, 1998.

[46] H. I. Jacobson. The maximum variance of restricted unimodal distributions. *The Annals of Mathematical Statistics*, 40(5):1746–1752, 1969.

[47] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[48] R. Axelrod. Effective choice in the prisoner's dilemma. *The Journal of Conflict Resolution*, 24(1):3–25, March 1980.

[49] R. Axelrod. More effective choice in the prisoner's dilemma. *The Journal of Conflict Resolution*, 24(3):379–403, September 1980.

[50] X. Yao and P. J. Darwen. How important is your reputation in a multi-agent environment. In *Proc. IEEE 1999 Conference on Systems, Man, and Cybernetics (SMC'99)*, pages 575–580, Tokyo, Japan, October 12–15, 1999.

[51] P. J. Darwen and X. Yao. Does extra genetic diversity maintain escalation in a co-evolutionary arms race. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 4(3):191–200, July 2000.

[52] D. B. Fogel. The evolution of intelligent decision making in gaming. *Cybernetics and Systems: An International Journal*, 22:223–236, 1991.

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

[54] D. B. Fogel. On the relationship between the duration of an encouter and the evolution of cooperation in the iterated prisoner's dilemma. *Evolutionary Computation*, 3(3):349–363, 1996.

[55] B. A. Julstrom. Effects of contest length and noise on reciprocal altruism, cooperation, and payoffs in the iterated prisoner's dilemma. In *Proc. 7th International Conf. on Genetic Algorithms (ICGA'97)*, pages 386–392, San Francisco, CA, 1997. Morgan Kauffman.

[56] R. Axelrod and W. D. Hamilton. The evolution of cooperation. *Science*, 211:1390–1396, 1981.

[57] P. G. Harrald and D. B. Fogel. Evolving continuous behaviors in the iterated prisoner's dilemma. *BioSystems: Special Issue on the Prisoner's Dilemma*, 37:135–145, 1996.

[58] N. Franken and A. P. Engelbrecht. Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma. *IEEE Transactions on Evolutionary Computation*, 9(6):562–579, 2005.

[59] M. A. Potter and K. A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.