Speciation as Automatic Categorical Modularization

Paul J. Darwen, *Associate Member, IEEE*, and Xin Yao, *Senior Member, IEEE*

***Abstract*— Real-world problems are often too difficult to be solved by a single monolithic system. Many natural and artificial systems use a modular approach to reduce the complexity of a set of subtasks while solving the overall problem satisfactorily. There are two distinct ways to do this. In *functional* modularization, the components perform very different tasks, such as subroutines of a large software project. In *categorical* modularization, the components perform different versions of basically the same task, such as antibodies in the immune system. This second aspect is the more natural for acquiring strategies in games of conflict.**

An evolutionary learning system is presented which follows this second approach to automatically create a repertoire of specialist strategies for a game-playing system. This relieves the human effort of deciding how to divide and specialize: species automatically form to deal with different high-quality potential opponents, and a gating algorithm manages the repertoire thus created. The genetic algorithm speciation method used is one based on fitness sharing. The learning task is to play the iterated prisoner's dilemma. The learning system outperforms the Tit-for-Tat strategy against unseen test opponents. It learns using a "black box" simulation, with minimal prior knowledge of the learning task.

***Index Terms*— Categorical modularization, co-evolution, evolutionary learning, implicit fitness sharing, iterated prisoner's dilemma, niching, speciation, tit-for-tat.**

I. Introduction

MANY problems in machine learning are large and complex. To reduce this complexity, a modular system uses specialized parts (or modules) to solve different aspects of the problem. Modular systems often provide suitable performance, e.g., separate parts of the brain deal with different aspects of vision [1], and artificial neural networks for vision emulate this modularity [2], [3].

Modular system design in engineering, however, has relied on human expertise (often a committee) to manually divide a system into specialized parts, often in an *ad hoc* manner. Also, manual modularization cannot be used on "black box problems" or "knowledge-lean problems" problems, for which little human expertise exists.

Instead, this paper uses speciation as a modularizing mechanism. Emulating speciation (or "niching") in evolutionary algorithms is not new and is widely used for multimodal function optimization. The original genetic algorithm (GA) suffers from genetic drift, causing its population to eventually cluster around only one solution, even if there are several equally attractive solutions in the search space. Speciation refers to any of the numerous modifications to the original GA that overcome this effect, allowing multiple solutions to be found.

This paper proposes and evaluates an evolutionary learning method for automatically designing a modular system. The novelty is in utilizing GA speciation as a modularizing mechanism: species automatically form to deal with separate parts of the problem, and this decomposition proceeds without human oversight. A gating algorithm manages the resulting repertoire of specialists. This approach learns using a "black box" simulation, with minimal *a priori* knowledge.

Manuscript received November 25, 1996; revised March 18, 1997 and April 18, 1997. This work was supported in part by a University College Postgraduate Research Scholarship (for P. J. Darwen) and by an Australian Research Council's Small Grant. This paper was presented in part at the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96), Nagoya, Japan, May 1996.

P. J. Darwen is with the DEMO Laboratory of the Computer Science Department, Brandeis University, Waltham, MA 02254-9110 USA (e-mail: darwen@cs.brandeis.edu.)

X. Yao is with the School of Computer Science, Australian Defence Force Academy, Canberra 2600 Australia (e-mail: xin@cs.adfa.oz.au).

Publisher Item Identifier S 1089-778X(97)05874-8.

II. Background

A. Two Types of Modularization and Generalization

One way to decompose a solution is to create "modules" that serve very different functions, not different versions of the same function. The top-down structure of large software projects is an example—each procedure has its own task. This goes by the label of *functional* modularization.

Another way is where the "modules" perform different versions of basically the same job. We call this *categorical* modularization. Antibodies in the immune system are an example (i.e., they all locate and destroy infection). The body must maintain a repertoire of antibodies, each specializing (to some extent) against a different type of antigen. The aim is to survive *any* infection.

This requirement, to provide some protection against any possible attack, is related to issues of *generalization* in learning. Strictly speaking, the immune system (like the learning system below) does not use supervised learning where generalization can be rigorously defined. This paper will abuse nomenclature and use "generalization" to mean how well a diverse repertoire of specialized experts cope with diverse, unseen opponents.

Categorical modularization is significant: many control, management, and military problems have requirements similar to the immune system. The simplest such problem is learning a game, such as baseball. Instead of creating a player to be good at everything, a more promising route is to decompose the solution into specialized parts—such as pitching, batting, and fielding—and deal with those aspects separately. Speciation can act as a modularizing mechanism to do this.

Payoff to player A

		Player B: C	Player B: D
Player A	C	2	0
	D	3	1

Fig. 1. Payoff to player A in the two-player prisoner's dilemma game. Each player can either cooperate (C) or defect (D).

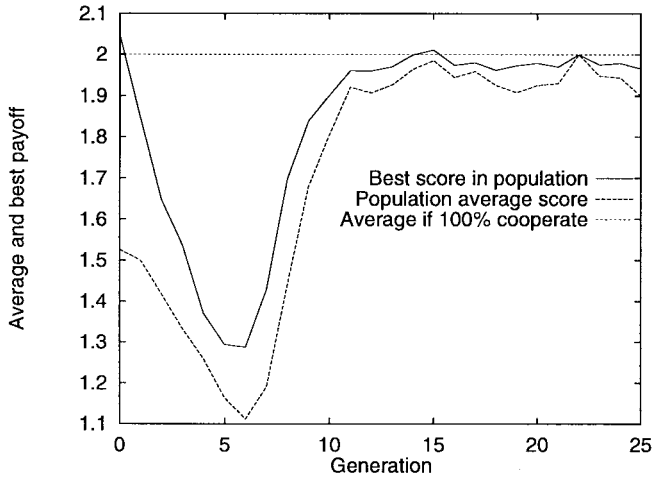

Fig. 2. A co-evolving nonspeciated GA learning the 2IPD. Starting from a random initial population, high-scoring cooperative strategies quickly dominate.

B. Problems with Co-Evolving Game Players

Co-evolution seems like an attractive learning method, especially for learning to play games. By continuously evolving novel answers to existing expertise, co-evolution should cause an escalating "arms race" of innovation [4, pp. 179–193].

The learning task studied here is the iterated prisoner's dilemma (IPD), a popular test for evolutionary learning [5]–[9]. Fig. 1 shows the relevant payoff matrix for the two-player prisoner's dilemma. The general definition of the prisoner's dilemma is available in [10, p. 91]. In a two-player iterated prisoner's dilemma (2IPD), the players proceed through many rounds of prisoner's dilemma, possibly remembering earlier actions.

Axelrod [5, p. 38] and others [6]–[8] showed that a GA, playing members of a population against each other in the 2IPD, could discover high-scoring strategies. Fig. 2 replicates a version of Axelrod's result, showing a typical run of a co-evolving GA learning a 2IPD: from a random initial population, the population is soon dominated by high-scoring cooperative strategies.

Unfortunately, this approach to co-evolutionary learning can give poor generalization ability. Genetic drift occurs in a GA when a finite evolving population clusters around a particular solution in the absence of selection pressure, due to variance in the selection process [11]. Genetic drift may cause the canonical GA to find only one solution, even if the search space offers multiple solutions [12, p. 185].

With co-evolution, genetic drift makes the GA over-specialize to a single strategy of the game. In the case of the IPD, the population soon consists of slight variations of the same strategy, usually a cooperative one [13]. This stalls the co-evolutionary arms race. Training against opponents which lack variety causes poor generalization [14] and makes those over-specialized individuals vulnerable to novel opponents [6], [15, p. 325]. Eventually, mutation produces a sufficiently novel opponent, which causes sudden mass extinctions [6]. Fig. 3 shows this for a nonspeciated co-evolutionary GA learning the IPD.[1]

Incidentally, collapses like that depicted in Fig. 3 in the IPD have been predicted by theory. For example, if a co-evolving population comprises more than 50% Tit-for-Tat[2] strategies, then all individuals will eventually use Tit-for-Tat or All-Cooperate and remain there *as long as* a sufficient number of Tit-for-Tats are maintained [17, pp. 121–122]. It has also been shown that Tit-for-Tat usually leads to even more cooperative strategies [13]. Fig. 3 shows what can occur when this over-specialization produces brittle strategies that can be invaded by defection.

As with practically all nontrivial games, the IPD does not offer a perfect[3] strategy [24]. The challenge is to learn a strategy that is as general as possible. Co-evolving nonbrittle strategies for games of conflict (even the IPD) is a challenging problem for evolutionary computation.

C. A Solution: Emulate Speciation

Speciation (or "niching") refers to various extensions to the canonical GA that allow it to find multiple optima. Speciation

[1] For Figs. 2 and 3, the population size was 50, elitism and single-point crossover was used, and the mutation rate was unusually high at 1.75 times the inverse of the genotype's length in bits. The reason mutation was set so high is that although collapses like in Fig. 3 occur over a wide range of mutation rates, they are quite rare. Increasing the mutation rate makes them less rare, without changing their character, to cause one as early as generation 100. A discussion of this phenomenon is available in [6].

[2] In the IPD, *Tit-for-Tat* is the name of the strategy that cooperates on the first iteration and then does whatever the opponent did on the previous iteration [16, p. 13]. This means it can achieve high scores against cooperative strategies, but cannot be heavily exploited by uncooperative opponents. It was the most successful strategy in a famous IPD tournament of computer programs (despite being the simplest) [16, ch. 2] and remains a useful standard for performance in the IPD.

[3] The IPD with noise offers an evolutionarily stable (ES) strategy [18]. However, the concept of evolutionary stability is (despite the name) *not* an accurate characterization of stability in co-evolution:

- examples abound of non-ES equilibria which are nonetheless local attractors and thus locally stable [19, p. 16], [20, p. 477], i.e., not every attractor is ES [21, p. 29];
- for discrete generations (like a GA often is), an ES point is not necessarily an attractor for the co-evolving system [22, p. 104], [17, p. 18].

Spectacular collapses like Fig. 3 have been observed in other co-evolutionary systems [8], [23]. It is interesting to note that Lindgren [8] looked at an IPD *with* noise (which offers an ES strategy [18]), and Fig. 3 uses an IPD *without* noise (which lacks an ES strategy [24]), but both produce similar mass extinctions. This demonstrates that the concept of evolutionary stability may not usefully predict the outcome of co-evolutionary learning. Collapses like Fig. 3 happen because of over-specialization to one strategy (unless a perfect one exists), not because ES points exist.

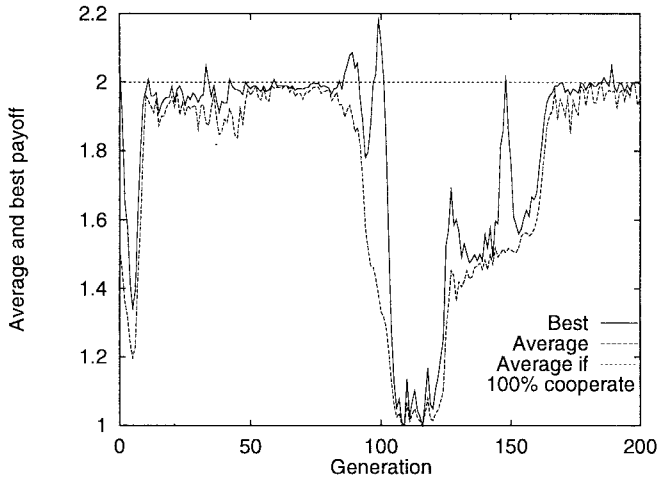

Fig. 3. A co-evolving GA learning the iterated prisoner's dilemma. After the GA becomes dominated by strategies that all behave similarly, the population over-specializes to the cooperative environment and becomes vulnerable to a new strategy that defects. Mutation eventually produces such a novel strategy, causing a mass extinction [6].

is a way to avoid a population of near-identical strategies and the resulting over-specialization and brittleness. The diverse strategies thus found are most effective only against certain opponents: each individual is not necessarily the best strategy against all possible opponents. Collectively, on the other hand, the diversity in these species can provide more versatile responses. The diverse expertise created by speciation is utilized by the gating algorithm to be described in Section III-C.

The various GA speciation methods are usually motivated by biological mechanisms of speciation. These include spreading individuals over a landscape, so that distance causes geographic mating restrictions [25], [26]. Mysteriously, this landscape approach maintains diverse IPD strategies only for particular ranges of the payoffs [9, p. 309].

D. Speciation with Implicit Fitness Sharing

Fitness sharing [27], [12, pp. 191–192] reduces the realized fitness at heavily populated regions of the search space to encourage search elsewhere. It has achieved spectacular results in difficult search problems [28], but also has nonnegligible flaws [29], [30].

An extension to fitness sharing, called "implicit fitness sharing" [30], [31], attempts to work around these flaws. Like the original, implicit fitness sharing also reduces the realized fitness at heavily populated regions of the search space. It does this with the algorithm in Table I. At first glance, Table I looks completely different from ordinary fitness sharing; however, they share the same theoretical basis [30].

Note that step 1 of Table I does not specify *how* to sample strategies from the population. This sampling is usually uniformly at random [30]. A recent study, however, found better results when sample members are chosen to give a diverse sample [32, p. 375]. Following [32], the current effort uses implicit fitness sharing both with the original random sampling and with *disassortative* sample selection: the first sample

TABLE I
PAYOFF FUNCTION FOR IMPLICIT FITNESS SHARING [30]

For each strategy i in the population, repeat the following C times:

1. From the population, select a sample of σ strategies.
2. Find the strategy in that sample which achieves the highest score[a] against the single test strategy i.
3. The best strategy in the sample receives a payoff. In the case of a tie, payoff is shared equally among the winners.

[a] Alternatively, the largest winning margin.

member is chosen randomly, and each successive member is the one most different (in terms of Hamming distance) from those already in the sample.

The modular approach studied here could use any of numerous speciation methods that work well in multimodal optimization [33], [34] and is not limited to implicit sharing.

E. Related Work

1) Building on Two Research Directions: Some studies have looked at co-evolution with speciation, but did not utilize the repertoire of expertise thus found. Smith and Gray [35] found diverse high-quality strategies in Othello using a co-evolutionary GA with speciation, but did not address the question of when to use which strategy. Similarly, Rosin and Belew [32] also used a speciated, co-evolutionary GA to learn game strategies. They hoped, however, that a single individual in the GA would be sufficiently well rounded to play against all possible opponents, instead of using more of the diverse GA population. The test games (other than Go) in [32] were simple enough for the GA to find a perfect strategy, side-stepping the question of when to use which specialized strategy.

Conversely, some studies [36]–[38] have *manually* divided a problem into various components and then applied a GA population to each part. This has been an effective method for the problems studied, but it requires some human knowledge on how to divide a monolithic problem into simpler pieces. Admittedly, deciding which does what piece can be relatively indirect—for example, Potter and De Jong [38] used a different population for each layer of a multilayer neural network and also seeded different populations with what the authors thought would be "a fairly natural decomposition" of the problem [37, p. 370]—but this still requires a person (or a committee) to decide how to construct this division.

This study combines and builds on these two research directions: a speciated GA decides autonomously how to specialize, and these parts are managed by a gating algorithm.

2) Cultural Algorithms: Other studies have considered the first approach mentioned in Section II-A, functional modularization. Randomly selected parts of evolving individuals are stored off-line and can be re-introduced to the population later [39]–[41].

This is similar to *cultural algorithms* [42]–[44] where the off-line cache for storing parts of evolving individuals (called

a "library" or "belief space") emulates observations that culture is varied and transmitted in ways analogous to genetic evolution [45]–[47, ch. 11], [48], an intriguing idea, both for machine learning and also to gain insights into how societies work.

Instead of *randomly* cutting and pasting parts of individuals into a library, here GA speciation is the automatic mechanism for dividing and specializing.

III. Method

A. Strategy Representation

Following Axelrod's representation [5], each individual is a set of rules, a binary string representing a look-up table that covers every possible history of the l most recent iterations of the IPD, where $l \leq 4$ steps. This was chosen because only recent steps will have significance for the next move.

A history of the l most recent rounds of the IPD consists of l bits for the player's own previous l moves, and another l bits for the other player's previous l moves. A "1" indicates defection and a "0" cooperation.

For example, let the number of remembered rounds be $l = 3$, and let one of the players perceive history as `001 111`. Here, the `001` indicates the player's own actions: the most recent action (on the left) was a "0," indicating cooperation, and the player's own action three steps ago (on the right), was a "1," i.e., defection. Similarly, the other player's previous actions are represented by `111`, three defections. Given this history, the first player would take the action listed in bit $001111_2 = 15$ of its genotype. The other player would view the same series of events as `111 001` and would take the action listed in bit $111001_2 = 57$ of its genotype. There are 2^{2l} possible histories of length l, so each individual's genotype must be that many bits in length so as to store an action for each possible history.

At the start of a game, there is no history of l rounds. A"pre-game" pseudo-history of another $2l$ bits specifies a strategy's initial move. Hence each strategy is finally represented by a binary string of length $2^{2l} + 2l$. For $l = 3$, this comes to 70 bits and $2^{70} \approx 10^{21}$ possible strategies.

Although the approach presented here uses rule sets to represent strategies, it could easily use other representations such as artificial neural networks or decision trees.

B. Genetic Algorithm Parameters

The parameters below were selected after a modest amount of investigation. The different parameters for $l = 4$ were chosen to keep running times reasonable. For each value of l, 30 GA runs started from different random initial populations.

- The population size was 50 for remembered history $l = 2, 3$, and 200 for $l = 4$.
- Mutation rate was one-fifth of the inverse of the genotype length.
- Following previous work [29, p. 168], [27, p. 49], [49] assortative crossover (where individuals cross over with similar individuals) is used instead of the usual random crossover, except that individuals do not crossover with identical copies of themselves (which has no effect).
- The crossover rate was at the maximum of one. The reason it was set so high is because assortative crossover reduces the disruptive effects of crossover.
- Selection was linear ranked selection (with the best in the population expecting two offspring and the worst no offspring) with simple elitism: the best individual always appeared in the next generation.
- From Table I, sample sizes were $\sigma = 0.2$ of the population for $l = 2, 3$ and $\sigma = 0.1$ of the population for $l = 4$. The number of samples for each test strategy was $C = 10$ for $l = 2, 3$ and $C = 5$ for $l = 4$.

Normally, an IPD game with a fixed number of iterations would cast the "shadow of the future" [16, p. 13]. Imagine a game is known to last for (say) 100 rounds, then on the final round there is no opportunity for the other player to retaliate to a defection, so you may as well defect. The other player is probably also thinking this and will most likely defect on the last round as well. So there is nothing to lose by defecting on the *second*-last round. And so on back to the start of the game. The real-life situations that the IPD models are open ended: nobody knows when the game will finish. This uncertainty, the "shadow of the future," is usually modeled in the IPD learning task (often by the "discount parameter" [16, p. 13]), but strategies used here have a representation that is too simple to be able to learn to utilize the fact that the number of rounds of the IPD is finite and fixed. So each game, both during the GA learning and against the test set, had a fixed duration of 100 iterations.

C. The Gating Algorithm

The final generation of a speciated GA contains diverse strategies. One could run another GA for each individual to search for the single best counter-strategy for that individual. The final GA population, however, presumably contains suitable counterstrategies to its own members. So a simple way to obtain "modules" (i.e., a diverse repertoire of specialized strategies to use against any possible opponent) is to merely copy the individuals in the last generation of the speciated GA. Although this is the least sophisticated way to extract a repertoire of diverse experts from a speciated GA, results in Section IV show that even this simple approach achieves a more versatile response against unseen opponents than the IPD strategy Tit-for-Tat or using the best-in-population strategy of the same GA population. More sophistication in this stage could give better performance in the future.

There are many ways to adjudicate among the resulting repertoire of strategies. Using an ensemble of experts is common for neural networks [50], [51]. The term *gating* algorithm is a generic description for an algorithm that chooses between different experts. Methods include:

- voting [50];
- model averaging [52];

- committees [53], [54];
- agent teams [55];
- stacked generalization [56], [57];
- Monte Carlo methods [58].

We used voting, the least sophisticated method of those listed. A more refined method, which could lead to improved performance, is left to future work.

To play against an online opponent, the gating algorithm must decide which strategy is best against the opponent's current strategy. The gating algorithm takes the speciated GA's final generation (which finished before meeting the online opponent) and proceeds as follows.

1) It reads in the final generation of the GA, excluding poor performers (the worst 25%).
2) It finds which individuals most closely resemble the opponent. From the most recent rounds of the game, the gating algorithm compares[4] what the opponent did with what each individual in the GA population would have done in the opponent's situation. Those GA individuals that most resemble the opponent are called the *imitators*.
3) Next, every individual plays against every imitator. These games start from the current history of moves against the actual opponent. As this needs to peer into the future only a short distance, these games are only for four iterations of the IPD.[5] Those that score the highest against the imitators are called the *imitator-answers*. This can be precalculated for greater speed, as the final population does not change during the test phase.
4) Finally, the imitator-answers vote on what action (cooperate or defect) to take in the current situation against the actual opponent.

In other words, the gating algorithm finds what strategy the online opponent is currently following and then finds the strategy which is specialized against the one the online opponent appears to be using.

This approach relies on speciation to find all the good strategies for the game. If the speciated GA has *not* found a reasonably complete ensemble, then the population will naïvely overfit to the strategies it knows about, causing vulnerability to unseen strategies. This is also true, however, for the best individual. If the speciated GA finds two or more different high-quality strategies for the game, then the modular approach should generalize better against diverse unseen opponents.

At the game's start, there are no previous moves to indicate the opponent's strategy. The gating algorithm takes a vote of the final GA generation for the presumed pregame moves (included in the genotype, described in Section III-B).

[4] A bitwise comparison, out of cooperate or defect.

[5] Four iterations gave the best performance after a modest amount of searching—with more than four iterations, performance plateaus (actually it declines slightly). It would have been interesting to see the effect of something more elaborate, such as having a weighted payoff for these short games so that distantly future iterations were less important. For tasks where looking further into the future carries a high computational cost, a more complicated decision function might be used to decide whether or not the larger computational effort justifies the extra steps into the future. This gets into nontrivial questions quite quickly, however, and such considerations can be left to the future.

TABLE II
RESULTS FOR REMEMBERED HISTORY $l = 2$. SEE TEXT FOR DETAILS

	$l = 2$				
Strategy	Wins	Ties	Losses	Average score	
	%	%	%	Own	Other's
best.ns	20.8	15.3	63.9	1.055	1.435
best.sr	34.5	31.2	34.3	1.274	1.164
gate.sr	50.7	6.3	43.0	1.301	1.139
best.sd	64.4	8.5	27.1	1.423	0.986
gate.sd	56.8	10.5	32.7	1.422	0.971
Tit-for-Tat	0.0	88.0	12.0	1.219	1.222

D. Unseen Test Strategies to Measure Generalization

Strategies acquired from the GA's final population were played against a set of unseen test opponents to test their generalization ability. As mentioned earlier, this is not reinforcement learning, so there is no rigorous way to define generalization ability. Unseen test opponents are a practical alternative, and as Wolpert [59, p. 246] observes, "It may turn out that ... there is no way to measure generalization more rigorously than via test problems."

One might argue that whatever test set is used, it somehow unfairly favors one or another of the methods being tested. To avoid any human favoritism, the test strategies were chosen from a large random sampling of all possible strategy genotypes, for each value of remembered history l. Each strategy sampled was evaluated against a large number of randomly chosen strategies (1000 for $l = 2, 3$, 10 000 for $l = 4$). Only the best 25 were kept for each set of unseen test strategies.

Choosing the test strategies this way means that there is no human input about what a "good" strategy ought to be. It also means that the strategies selected perform acceptably against a broad range of opponents, i.e., they are not brittle.

IV. RESULTS

Tables II–IV show the performances against the unseen test opponents for 30 of each of the following strategies.

- The best single individual from the final generation of a co-evolutionary GA without speciation (labeled "best.ns" in Tables II–IV), following Axelrod [5].
- A co-evolutionary GA using implicit fitness sharing with *random* sample selection in step 1 of Table I, using the best individual from the final generation (labeled "best.sr"), and the gating algorithm on the entire final generation (gate.sr).
- A co-evolutionary GA using implicit fitness sharing with *disassortative* sample selection,[6] using the best individual from the final generation (best.sd), and the gating algorithm on the entire final generation (gate.sd). With the larger population used for $l = 4$, disassortative sample selection was too slow to be practical.

There are 30 trials of each value of l, and 25 unseen test opponents for each value of l, so each line in Tables II–IV is

[6] Where the next individual chosen for the sample is the one least like those already in it, as measured by Hamming distance, for step 1 of Table I. The aim is to get more diverse samples, following an earlier study [32].

TABLE III
RESULTS FOR REMEMBERED HISTORY $l = 3$

	$l = 3$				
Strategy	Wins	Ties	Losses	Average score	
	%	%	%	Own	Other's
best.ns	18.5	5.1	76.4	1.016	1.458
best.sr	18.3	8.0	73.7	1.039	1.452
gate.sr	25.9	5.3	68.8	1.147	1.347
best.sd	19.1	8.9	72.0	1.006	1.470
gate.sd	39.1	11.3	49.6	1.291	1.125
Tit-for-Tat	0.0	32.0	68.0	1.193	1.213

TABLE IV
RESULTS FOR REMEMBERED HISTORY $l = 4$. DISASSORTATIVE SAMPLING (.sd) WAS NOT USED BECAUSE THE LARGER POPULATION MADE IT COMPUTATIONALLY EXPENSIVE

	$l = 4$				
Strategy	Wins	Ties	Losses	Average score	
	%	%	%	Own	Other's
best.ns	34.3	5.7	60.0	1.235	1.595
best.sr	36.0	5.9	58.1	1.322	1.513
gate.sr	64.3	5.9	29.8	1.520	1.234
Tit-for-Tat	0.0	44.0	56.0	1.431	1.448

the average of 750 games of the IPD. Instead of 30, there is only one Tit-for-Tat, so its average is only out of the 25 games against the unseen test opponents.

V. DISCUSSION

A. Criteria for Judging Methods

The aim is for co-evolution to produce all-purpose expertise, instead of a brittle and over-specialized solution. As described in Section II-A, however, there is no rigorous measure of generalization ability against unseen opponents, so the GA cannot use such a measure as its fitness. Instead, the fitness criterion is simply to maximize game scores.

But, like an immune system, the criterion for strategies' generalization ability is how well they protect against a test set of unseen opponents. For this reason, Tables II–IV show not only the average game scores of the various methods, but also the number of games won, tied, and lost. So a strategy generalizes "better" than another if it both achieves a higher score and suffers fewer losses.

Considering both the absolute game scores and the number of losses as success criteria is unusual in this domain. For example, in Axelrod's original tournaments [16, ch. 2], the interesting result was that the Tit-for-Tat strategy was able to win the overall tournament, despite ties or narrow losses in most games. Consistently losing by a small amount is not typical of naturally evolved organisms, nor should co-evolutionary learning aim so low.

B. Comparisons

For the simplest 2IPD with history $l = 2$, where strategies are only 20 bits in length, methods using speciation give similar performance in Table II. Closer inspection of the 25 unseen test strategies for $l = 2$ reveals low diversity—they are all basically the same strategy. This is because the method of finding test sets (Section III-D) found little diversity in the small space of strategies with $l = 2$. Against these nondiverse opponents, a modular approach gives little advantage.

It is worth noting that in a recent study [32], all the test games (except Go) were simple enough for the speciated, co-evolutionary GA to find a perfect strategy. For such simple games, or (like in Table II) where possible opponents lack variety, a modular approach is superfluous.

As the space of strategies becomes larger with $l > 2$, the gated strategy gains a winning margin. In Tables III and IV, the gated strategies achieve higher scores and suffer fewer losses than using the best-in-population. This superiority is robust to variations in the actual speciation method—with either random sample selection or disassortative sample selection, the gated strategy is still better. This validates the approach of speciation as a modularizing mechanism.

C. Tit-for-Tat

As expected, Tit-for-Tat ties with, or narrowly loses to, the unseen test opponents in Tables II–IV.

But in Table III, Tit-for-Tat is outperformed by the gated strategy when it uses the final population using implicit fitness sharing with *disassortative* sample selection (labeled "gate.sd"). That is, Tit-for-Tat was worse both in the number of losses, and in the score. In Table IV, Tit-for-Tat again suffers more losses and achieves a lower score than the gated strategy.

Versatility against unseen opponents is a desirable attribute for a machine learning system. Tit-for-Tat has long been known as an IPD strategy that performs well against a wide range of opponents [16, ch. 2]. In contrast, the gated strategy can bring to bear more versatile responses and provide better protection against unseen opponents. That the gated strategy compares well with Tit-for-Tat lends credibility to the modular approach.

D. The Need for Better GA Speciation

From step 1 of Table I, the speciation method can sample either randomly or with disassortative sampling. In both Tables II and III, disassortative sampling (labeled "gate.sd") outperforms random sampling (labeled "gate.sr"). That is, both the number of losses and average score are worse for random sampling. In Table III, improved sampling is what tilts the balance to outperform Tit-for-Tat.

This difference indicates that using speciation as a modularizing mechanism relies heavily on the speciated GA to maintain diversity and works better as the speciation method also works better.

Disassortative sampling is a minor and fairly *ad hoc* improvement to the speciation method, and its computational expense makes it impractical for the larger population used for $l = 4$ in Table IV (in which even random sampling allowed the gated strategy to beat Tit-for-Tat). Nonetheless, it improves the performance of the modular approach. How much better, then, could a well-thought-out GA speciation method be? These

results indicate the need for improved speciation methods in evolutionary computation.

VI. Conclusions

Using co-evolution to learn to play games can give poor generalization ability. A canonical GA often converges to only one solution at a time, and with co-evolution it over-specializes to one particular strategy, becoming vulnerable to novel opponents. This brittleness can cause mass extinctions as in Fig. 3, when mutation eventually produces such a novel strategy [6].

Some studies have nonetheless used both speciation and co-evolution [32], [35]. This prevents over-specialization and improves generalization ability. Although these approaches learned diverse and specialized strategies, they did not address the question of when to use which species. This wastes much of the specialized expertise found by speciation.

Speciation can act as a modularizing mechanism, and existing gating methods can utilize the diverse expertise embodied in the species thus found. This approach automatically creates a modularized solution to the problem of finding a good strategy to a game, without human teachers or prior knowledge. It uses only a "black box" simulation of the game itself. Against a test set of unseen opponents, chosen in a way that removed human bias, this method outperforms a known high-quality strategy.

Acknowledgment

The authors would like to thank D. B. Fogel and the anonymous referees for encouragement and helpful comments.

References

[1] D. Grady, "The vision thing: Mainly in the brain," *Discover*, vol. 14, pp. 57–66, June 1993.
[2] T. C. Folsom, "A modular hierarchical neural network for machine vision," in *Proc. Int. Joint Conf. Neural Networks*, 1990, pp. 897–902.
[3] M. M. V. Hulle and G. A. Orban, "The EDANN concept: A modular artificial neural network model for biological vision and image processing," in *Proc. World Congr. Neural Networks*, vol. 4. Hillsdale, NJ: Lawrence Erlbaum, p. 320, 1994.
[4] R. Dawkins, *The Blind Watchmaker*, 1st ed. Harlow, U.K.: Longman, 1986.
[5] R. M. Axelrod, "The evolution of strategies in the iterated prisoner's dilemma," in *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed. Los Altos, CA: Morgan Kaufmann, 1987, pp. 32–41.
[6] P. J. Darwen and X. Yao, "On evolving robust strategies for iterated prisoner's dilemma," in *Progress in Evolutionary Computation*, (Lecture Notes in Artificial Intelligence, vol. 956) X. Yao, Ed. Berlin: Springer, 1995, pp. 276–292.
[7] D. B. Fogel, "Evolving behaviors in the iterated prisoner's dilemma," *Evolutionary Computation*, vol. 1, no. 1, pp. 77–97, 1993.
[8] K. Lindgren, "Evolutionary phenomena in simple dynamics," in *Artificial Life 2*, (Santa Fe Institute Studies in the Sciences of Complexity, vol. 10) C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Eds. Redwood City, CA: Addison-Wesley, 1991, pp. 295–312.
[9] ———, "Evolutionary dynamics of spatial games," *Physica D*, vol. 75, pp. 292–309, 1994.
[10] X. Yao and P. J. Darwen, "An experimental study of N-person iterated prisoner's dilemma games," in *Progress in Evolutionary Computation*, (Lecture Notes in Artificial Intelligence) X. Yao, Ed., vol. 956. Berlin: Springer, 1995, pp. 90–108.
[11] S. W. Mahfoud, "Genetic drift in sharing methods," in *Proc. First IEEE Conf. Evolutionary Computation*. June 1994, vol. 1, pp. 67–72.
[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[13] M. A. Nowak and K. Sigmund, "Tit for tat in heterogeneous populations," *Nature*, vol. 355, pp. 250–253, Jan. 16, 1992.
[14] S. L. Epstein, "Toward an ideal trainer," *Machine Learning*, vol. 15, pp. 251–277, 1994.
[15] T. Ikegami, "From genetic evolution to emergence of game strategies," *Physica D*, vol. 75, pp. 310–327, 1994.
[16] R. M. Axelrod, *The Evolution of Cooperation*. New York: Basic Books, 1984.
[17] R. Cressman, *The Stability Concept of Evolutionary Game Theory* (Lecture Notes in Biomathematics, vol. 94). Berlin: Springer-Verlag, 1992.
[18] R. Boyd, "Mistakes allow evolutionary stability in the repeated prisoner's dilemma game," *J. Theoretical Biology*, vol. 136, pp. 47–56, 1989.
[19] J. Hofbauer and K. Sigmund, *The Theory of Evolution and Dynamical Systems* (London Mathematical Society Student Texts, vol. 7). Cambridge, U.K.: Cambridge Univ. Press, 1988.
[20] E. C. Zeeman, "Population dynamics from game theory," in *Global Theory of Dynamical Systems*, Z. Nitecki and C. Robinson, Eds. (Lecture Notes in Mathematics, vol. 819) Berlin: Springer-Verlag, 1980, pp. 471–497.
[21] H.-M. Voigt, *Evolution and Optimization*. Berlin: Akademie-Verlag, 1989.
[22] C. Cannings, "Topics in the theory of evolutionarily stable strategies," in *Mathematical and Statistical Developments of Evolutionary Theory: Proceedings of the NATO Advanced Study Institute and Séminaire de Mathématiques Supérieures on Mathematical and Statistical Developments of Evolutionary Theory*, S. Lessard, Ed. Boston: Kluwer Academic, 1987, pp. 95–119.
[23] R. G. Palmer, W. B. Arthur, J. H. Holland, B. LeBaron, and P. Tayler, "Artificial economic life: A simple model of a stockmarket," *Physica D*, vol. 75, pp. 264–274, 1994.
[24] R. Boyd and J. P. Lorberbaum, "No pure strategy is evolutionarily stable in the repeated prisoner's dilemma game," *Nature*, vol. 327, pp. 58–59, May 7, 1987.
[25] N. Adachi and K. Matsuo, "Ecological dynamics under different selection rules in distributed and iterated prisoner's dilemma game," in *Parallel Problem Solving from Nature: First workshop*, H.-P. Schwefel and R. Männer, Eds. (Lecture Notes in Computer Science, vol. 496). Berlin: Springer-Verlag, 1990, pp. 388–394.
[26] H. Mühlenbein, "Darwin's continent cycle theory and its simulation by the prisoner's dilemma," *Complex Syst.*, vol. 5, pp. 459–478, 1991.
[27] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proc. Third Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 42–50.
[28] D. E. Goldberg, K. Deb, and J. Horn, "Massive multimodality, deception, and genetic algorithms," in *Parallel Problem Solving from Nature 2*, R. Männer and B. Manderick, Eds. New York: North-Holland, 1992, pp. 37–46.
[29] P. J. Darwen and X. Yao, "A dilemma for fitness sharing with a scaling function," in *Proc. 1995 IEEE Conf. Evolutionary Computation*, 1995, pp. 166–171.
[30] R. E. Smith, S. Forrest, and A. S. Perelson, "Searching for diverse, cooperative populations with genetic algorithms," *Evolutionary Computation*, vol. 1, no. 2, pp. 127–149, 1992.
[31] S. Forrest, B. Javornik, R. E. Smith, and A. S. Perelson, "Using genetic algorithms to explore pattern recognition in the immune system," *Evolutionary Computation*, vol. 1, no. 3, pp. 191–211, 1993.
[32] C. D. Rosin and R. K. Belew, "Methods for competitive co-evolution: Finding opponents worth beating," in *Proc. Sixth Int. Conf. Genetic Algorithms*, L. J. Eshelman, Ed. San Mateo, CA: Morgan Kaufmann, 1995, pp. 373–380.
[33] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, Univ. of Illinois at Urbana-Champaign, 1995.
[34] S. Ronald, "Finding multiple solutions with an evolutionary algorithm," in *Proc. 1995 IEEE Conf. Evolutionary Computation*, pp. 641–646.
[35] R. E. Smith and B. Gray, "Co-adaptive genetic algorithms: An example in Othello strategy," in *Proc. 1994 Florida Artificial Intelligence Research Symp.*, D. D. Dankel, Ed., Saint Petersburg, FL, 1994, pp. 259–264.
[36] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," in *Artificial Life 2*, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Eds. (Santa Fe Institute Studies in the Sciences of Complexity, vol. 10). Redwood City, CA: Addison-Wesley, 1991, pp. 313–323.
[37] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Third Parallel Problem Solving*

from Nature. Berlin: Springer-Verlag, 1994, pp. 249–257.
[38] M. A. Potter, K. A. De Jong, and J. J. Grefenstette, "A coevolutionary approach to learning sequential decision rules," in *Proc. Sixth Int. Conf. Genetic Algorithms*, L. J. Eshelman, Ed. San Mateo, CA: Morgan Kaufmann, 1995, pp. 366–372.
[39] P. J. Angeline and J. B. Pollack, "Evolutionary module acquisition," in *Proc. 2nd Annu. Conf. Evolutionary Programming*, D. B. Fogel and W. Atmar, Eds. San Diego, CA: Evolutionary Programming Soc., Feb. 1993, pp. 154–163.
[40] K. E. Kinnear, "Alternatives in automatic function definition," in *Advances in Genetic Programming*, K. E. Kinnear, Ed. Cambridge, MA: MIT Press, 1994, pp. 119–142.
[41] J. R. Koza, *Genetic Programming 2: Automatic Discovery of Reusable Programs*. Cambridge, MA: MIT Press, 1994.
[42] R. K. Belew, "Evolution, learning and culture: Computational metaphors for adaptive search," *Complex Syst.*, vol. 4, no. 1, pp. 11–49, 1990.
[43] E. Hutchins and B. Hazlehurst, "Learning in the cultural process," in *Artificial Life 2*, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Eds. Redwood City, CA: Addison-Wesley, 1991, pp. 689–706.
[44] R. G. Reynolds and S. R. Rolnick, "Cultural algorithms," in *Proc. 1995 IEEE Conf. Evolutionary Computation*, 1995, pp. 819–824.
[45] R. Boyd and P. J. Richerson, *Culture and the Evolutionary Process*. Chicago, IL: Univ. of Chicago Press, 1985.
[46] L. L. Cavalli-Sforza and M. W. Feldman, *Cultural Transmission and Evolution: A Quantitative Approach* (Monographs in Population Biology, vol. 16). Princeton, NJ: Princeton Univ. Press, 1981.
[47] R. Dawkins, *The Selfish Gene*, 2nd ed. Oxford, U.K.: Oxford Univ. Press, 1989.
[48] M. I. Sereno, "Four analogies between biological and cultural/linguistic evolution," *J. Theoretical Biology*, vol. 151, no. 4, pp. 467–508, Aug. 1991.
[49] B. L. Miller and M. J. Shaw, "Genetic algorithms with dynamic niche sharing for multimodal function optimization," Illinois Genetic Algorithms Lab., Univ. of Illinois at Urbana-Champaign, Tech. Rep. 95010, Dec. 1995.
[50] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 10, pp. 993–1001, 1990.
[51] C. L. Scofield, L. Kenton, and J.-C. Chang, "Multiple neural net architectures for character recognition," in *Proc. Compcon 91: 36th IEEE Computer Society Int. Conf.*, 1991, pp. 487–491.
[52] W. L. Buntine and A. S. Weigend, "Bayesian back-propagation," *Complex Syst.*, vol. 5, no. 6, pp. 603–643, 1991.
[53] D. J. C. Mackay, "Bayesian methods for supervised neural networks," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA: MIT Press, 1995, pp. 144–148.
[54] H. H. Thodberg, "Review of Bayesian neural networks with an application to near infrared spectroscopy," *IEEE Trans. Neural Networks*, vol. 7, no. 1, pp. 56–72, 1996.
[55] U. Beyer and F. J. Smieja, "Learning from examples, agent teams and the concept of reflection," Institute for System Design Technology (GMD), Adaptive Systems group, Sankt Augustin, Germany, Tech. Rep. REFLEX 1993/1, GMD Rep. 776, June 1993.
[56] T. M. English, "Stacked generalization and simulated evolution," *BioSystems*, vol. 39, no. 1, pp. 3, 1996.
[57] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.
[58] R. M. Neal, "Probabilistic inference using Markov chain Monte Carlo methods," Dept. Comput. Sci., Univ. of Toronto, Tech. Rep. CRG-TR-93-1, 1993.
[59] D. H. Wolpert, "A mathematical theory of generalization," *Complex Syst.*, vol. 4, pp. 151–249, 1990.

Paul J. Darwen (S'96–A'96) received the B.Sc. degree in mathematics from the Australian National University, Canberra, Australia, in 1993 and the the Ph.D. degree in computer science from the University College of the University of New South Wales in 1996.

Since October 1996 he has been a Postdoctoral Research Fellow at the DEMO Laboratory of the Computer Science Department, part of the Volen National Center for Complex Systems at Brandeis University, Waltham, MA. His research interests are in co-evolutionary learning and its application to games of conflict.

Xin Yao (M'91–SM'96) received the B.Sc. degree from the University of Science and Technology of China (USTC), in Hefei, Anhui province, China, in 1982, the M.Sc. degree from the North China Institute of Computing Technologies, Beijing, in 1985, and the Ph.D. degree from USTC in 1990.

From 1985 to 1990 he lectured at USTC and subsequently held postdoctoral fellowships at the Australian National University in Canberra, Australia, and the Commonwealth Scientific and Industrial Research Organization Division of Building, Construction and Engineering in Melbourne, Australia. In September 1992 he accepted a faculty position at the School of Computer Science, University College, the University of New South Wales, Australian Defence Force Academy in Canberra, Australia. He was promoted to Senior Lecturer in July 1995. His research interests include evolutionary computation (especially evolutionary learning and optimization), artificial neural networks, simulated annealing, computational complexity, and data mining.

Dr. Yao is involved in many international conferences, including a Program Committee Co-Chair for IEEE ICEC'97 in Indianapolis, IN, and a Program Committee Co-Vice-Chair for IEEE ICEC'98 in Anchorage, AK.